

---

# **shallot - Manual**

***Release 1.2.4767***

**Josef Hahn**

**Jul 26, 2020**



# CONTENTS

<b>1</b>	<b>License</b>	<b>1</b>
<b>2</b>	<b>About</b>	<b>3</b>
<b>3</b>	<b>Up-to-date?</b>	<b>5</b>
<b>4</b>	<b>Maturity</b>	<b>7</b>
<b>5</b>	<b>Dependencies</b>	<b>9</b>
<b>6</b>	<b>Using Shallot</b>	<b>11</b>
6.1	Beginning . . . . .	11
6.2	Customization . . . . .	11
6.3	Plugin Interface . . . . .	13
<b>7</b>	<b>Appendix: Web Mode</b>	<b>15</b>
<b>8</b>	<b>Appendix: Installing A Shallot Plugin</b>	<b>17</b>
<b>9</b>	<b>Appendix: Installation</b>	<b>19</b>
9.1	Source Code Archive . . . . .	19
<b>10</b>	<b>API reference</b>	<b>21</b>
10.1	Class list . . . . .	21
10.2	Namespace list . . . . .	598
10.3	Struct list . . . . .	1880
	<b>Index</b>	<b>1889</b>



## LICENSE

shallot is written by Josef Hahn under the terms of the GPLv3 or higher.

Please read the *LICENSE* file from the package and the *Dependencies* section for included third-party stuff.



## ABOUT

Shallot is a file manager with the maximum degree of flexibility and customizability.

Some features:

- Unlimited number of file panels
- Lots of customization options, e.g.
  - directory tree can be hidden and is flexible in behavior
  - lots of view options
  - program behavior
  - ...
  - while conserving a good usability
- It nicely works with convoluted file systems, like editing an image in an archive in another archive on a network drive.
- It has a plugin interface which allows one to implement many additional functionality (like new filesystems) with Python scripting, without any compiler hassles.
- It is Qt5 based and plays nicely with modern Linux desktops. See the downloads which other operating systems are supported (maybe with a few slight limitations).





## UP-TO-DATE?

Are you currently reading from another source than the homepage? Are you in doubt if that place is up-to-date? If yes, you should visit <https://pseudopolis.eu/wiki/pino/projs/shallot> and check that. You are currently reading the manual for version 1.2.4767.



**MATURITY**

shallot is in production-stable state.



## DEPENDENCIES

There are external parts that are used by shallot. Many thanks to the projects and all participants.



*Typical GNU/Linux Desktop*, recommended



*Qt5*, required



*Python 3.4*, required



*Poco::Net (for web ui)*, optional



*Glib (for gio)*, optional



*libsecret*, optional



*font 'Symbola'*, included : for logo symbol; free for use; copied from [here](#).



*banner image*, included : `_meta/background.png`; license [CC BY-SA 3.0](#); copied from [here](#).



*icon set 'Elegant Font'*, included : files `'icons/from.elegant_font.'`; license [CC BY 3.0](#); see [homepage](#).



*all files in `_meta`*, included : if not mentioned otherwise, Copyright 2015 Josef Hahn under license [CC BY-SA 3.0](#) license.



## USING SHALLOT

### 6.1 Beginning

Please read how to make Shallot ready for the first steps in `h_installation`.

Whenever you start Shallot, you see the main window, which is assembled by the following parts:

- An action bar on the top: A lot of buttons which can trigger some actions. The main menu is also here ('three horizontal bars' icon).
- The address bar: See the current directory path and navigate upwards to the breadcrumbs or enter a new location.
- The directory tree on the left: A tree view of the filesystem(s) for easy navigation in wild directory hierarchies.
- The file view on the right: Shows the content of the current directory. There might be more than just one of this views (if you configure it this way).
- A status bar on the bottom: Comes up when it wants to tell you some things.

The general usage is not very different from other general purpose file managers. This text will not go that deep into it:

Navigate to filesystem places where you plan to execute some work, select files and apply some operations to them. The action bar shows some available operations, while the context menu will list some more.

Please use everything with caution according to the value of your data. You would not be the first one destroying a month of work by carelessly removing stuff. :-P

### 6.2 Customization

The Shallot user interface provides customization options at many places. Many view-related ones are in the view menu (what you can find in the action bar). There are of course many other places which provide some degree of customized behavior as well. All relevant ones can be conserved between sessions. There are two different categories of customization options in Shallot, which can be applied in a durable way:

- **Settings:** This category contains everything which is designed to routinely be changed by the user and which can be stored. This can e.g. be a view aspect like the visibility of the directory tree.
- *Under The Hood* stuff: Some options which work just fine when completely untouched for the most users. If you ever make a change here, this will be very rare situations. The path to some internal system tools can be considered here, if your system provides them in an untypical location.

Now some details will follow about these categories.

## 6.2.1 Settings

Settings can be done at many different places. While the view menu contains some actions for changing view-related settings, actions in other menus may somehow set settings as well. However, all those settings can be stored in a similar way. They all have another thing in common: They *must* be stored explicitly by the user, otherwise they will be dropped when Shallot closes.

### Saving Settings

For saving some settings, use ‘Save settings’ in the main menu.

The appearing dialog is quite powerful (as the Shallot settings system is). However, it has a reduced beginners mode, which appears at first.

In this mode, you can (un)mark some settings from the list to be saved for later sessions. This is quite simple. If you save, you have conserved the chosen settings together with the displayed values for the future. Each box represents one setting, the boxes are split up into categories for better overview.

You can do that many times, which accumulates all your settings internally. To unset some settings again, use the manage dialog as described in the next section.

In the header, there is a button for showing also advanced stuff. This brings a lot of more power. The following describes the additional functionality.

While the basic mode just allows storage independently of the directory, the advanced mode allows to bind settings to a directory. Whenever you visit this directory, the stored setting values become applied. In this regard you can also decide to apply just on that single directory or on all its subdirectories as well.

The multi-profile functionality is part of the advanced stuff as well. Instead of the default profile, you can split settings up into different profiles for different usage scenarios. The main menu allows to switch between profiles. In this regard you can also decide to also inherit settings from other profiles (either on the global or on the per-directory level).

Within some of the settings boxes, there now also appears a new check box. If you enable it, Shallot will bind this value just to a single file list. This is useful for storing different settings for different file views (if you use more than one). Since the dialog always refers to that file list, which is currently active in the main window, you have to use it more than once for storing different settings for different fileviews.

Finally there are some more settings (i.e. more boxes) available in the advanced mode.

### Managing Settings

The settings are accumulated internally in a last-one-wins ways at evaluation time. In storage, they reside mostly as you stored them. There are some reasons for managing the settings which are currently stored:

- getting an overview
- removing wrong or outdated stuff
- restructuring/changing/rebuilding

Use the ‘Manage saved settings’ action from the main menu for this concerns.

You should at first choose the correct profile if you are not interested in the selected profile but a different one. If you have never created a new Shallot profile, leave this as it is.

On the left side there is a list of all the moments you stored some settings. You see if they are globally or if they apply to a certain directory. You can remove single entries of that list after selecting it, or you can drop the entire profile. Selecting an entry shows more information.

On the right side, you see the content of the selected entry.



---

**Note:** You can't technically change things once they are stored. For doing this, you would have to close Shallot and directly modify some xml files somewhere deeply hidden in your user profile. This is actually the only option you have when everything else is not an option :-P However, the recommended procedure is to just remove the old entry and maybe to store the setting again according to your new wishes.

---

## 6.2.2 Under The Hood Stuff

All those options are accessible from one the 'Tuning' dialog. You find it in the main menu.

Each available option is one box in those dialog. A box contains a description text and the current value of that option. For better overview, the boxes are aligned in some tabs. Follow the on-screen instructions for changing something here.

## 6.3 Plugin Interface

There is a plugin interface for shallot, which allows external persons to develop additional features for Shallot. Those plugins are written in Python and use the Shallot scripting api for interaction.

### 6.3.1 Developing Plugins

For the beginning, you should enable the Plugin Management in Shallot. In the 'Tuning' dialog, you should find it.

After restarting, you find a new top-level node which give a list of installed plugins. It is divided into plugins built into Shallot itself, plugins installed somewhere for all users and plugins installed somewhere for just a the current user.

There is a comprehensive documentation for all further steps accessible from within the Plugin Management.



## APPENDIX: WEB MODE

There is a way to run Shallot as web application. This way it doesn't provide a native desktop window, but a web server which a web browser can connect to.

For web mode, run Shallot with the following command line:

```
shallot --ui web
```



## APPENDIX: INSTALLING A SHALLOT PLUGIN

If a plugin comes as an installation package for your operating system, just install it in the usual way. If it is the naked plugin file, copy the file into `$HOME/.shallot/plugins`.

Restart Shallot afterwards.



## APPENDIX: INSTALLATION

Install Shallot via the installation package for your environment, if a suitable one exists for download. This also takes care of installing dependencies and doing preparation (unless mentioned otherwise in the installation procedure). After the installation, you can skip the rest of this section.

### 9.1 Source Code Archive

Use the source code archive as fallback. Extract it to a location which is convenient to you (Windows users need an external archive program; for example the great ‘7-Zip’ tool). Also take a look at the [Dependencies](#) for external stuff you need to install as well.

After extraction, open the Shallot root directory in a terminal and run:

```
qmake  
make
```

This builds Shallot from the source code, resulting in an executable. You should be able to execute the new *shallot* executable afterwards.

Instead of the command line approach, you can also use the QtCreator IDE and just load and run it there.

---

**Note:** There are some configuration flags in the beginning of the *shallot.pro* file. You should check them before building and enable the optional features as desired.

---





## API REFERENCE

### 10.1 Class list

#### 10.1.1 Class `sh::MainInit`

**class** `sh::MainInit`  
(Un)Initialization for main app.

##### Public Static Functions

void **doInitialize** ()

void **doShutdown** ()

##### Public Static Attributes

`sh::base::SingletonInitializer` \***singletonInitializer** = nullptr

sig\_atomic\_t **caughtSigterm** = 0

#### 10.1.2 Class `sh::actions::AbstractActionFactory`

**class** `sh::actions::AbstractActionFactory`

Abstract base class for an action factory.

It creates an instance of a certain subclass of *AbstractActionItem*. It can also check if it is defined to be created in a given situation (e.g. not all actions are visible in toolbar).

Subclassed by `sh::actions::ActionFactory< T >`, `sh::scripting::api::ApiActionFactory`

##### Public Functions

**AbstractActionFactory** (std::shared\_ptr<`sh::actions::ActionCategory`> *category*,  
QList<std::shared\_ptr<*Predicate*>> *predicates*)

Is (for subclasses) intended to be directly constructed and registered once.

**~AbstractActionFactory** ()

std::shared\_ptr<*ActionInstantiation*> **actionAvailable** (*ActionInstantiation* \**instantiation*)

std::shared\_ptr<`sh::actions::AbstractActionItem`> **construct** (QList<std::shared\_ptr<`sh::filesystem::FilesystemNode`>>  
*nodes*) = 0

## Private Members

`std::shared_ptr<sh::actions::ActionCategory> category`

`QList<std::shared_ptr<Predicate>> predicates`

### 10.1.3 Class `sh::actions::AbstractActionItem`

**class** `sh::actions::AbstractActionItem` : **public** `QObject`, **public** `std::enable_shared_from_this<AbstractActionItem>`

Abstract base class for executable actions, submenus of them, and more.

See the subclasses of this class.

They are used at various places. The toolbar and context menus provide them to the user.

They can be registered e.g. in a `sh::actions::ActionFactory` or returned from `sh::filesystem::FilesystemHandler.getActions` in order to make them available.

Subclassed by `sh::actions::ActionActionItem`, `sh::actions::HeaderActionItem`, `sh::actions::SeparatorActionItem`, `sh::actions::SubmenuItem`

## Public Functions

`QString text ()`

Returns the displayed text for this action.

`QString icon ()`

Returns the icon for this action.

`bool enabled ()`

Checks if this action is enabled.

`bool isChecked ()`

Checks if this action is checked (has a cross in the ui).

`bool isCheckable ()`

Checks if this action is checkable (can have a cross in the ui).

`int defaultActionPrecedence () const`

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

`void setText (QString text)`

Sets the displayed text.

`void setIcon (QString icon)`

Sets the icon.

`void setEnabled (bool enabled)`

Sets if the item is enabled.

`void setChecked (bool checked)`

Sets if the item is checked (has a cross in the ui).

`void setVisible (bool visible)`

Sets the visibility of this item.

```

bool visible ()
    Checks the visibility of this item (non-recursively).

std::weak_ptr<AbstractActionItem> parentAction ()
    Returns the parent action, if it is added to a container.

void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.

AbstractActionItem ()

```

## Signals

```

void changed ()
    Emits when some data changed.

```

## Private Members

```

QString _text
QString _icon
bool _enabled
bool _visible
bool _ischeckable
bool _ischecked
bool _initialized
bool _initializing
bool _reinitialize
std::weak_ptr<AbstractActionItem> _parentAction = NO_PARENT
int _defaultActionPrecedence

```

## Private Static Attributes

`QWaitCondition` `initcondition`

### 10.1.4 Class `sh::actions::ActionActionItem`

**class** `sh::actions::ActionActionItem` : **public** `sh::actions::AbstractActionItem`

Abstract base class for an executable action.

It can be made visible in menus or the toolbar or executed directly.

Subclasses provide the actual implementation by implementing `action`. Execution can be triggered from outside by calling `execute`.

Subclassed by `sh::actions::common::ActionAbstractTransferTree`, `sh::actions::common::ActionAddToBookmarks`,  
`sh::actions::common::ActionBookmark`, `sh::actions::common::ActionClipboardCopy`,  
`sh::actions::common::ActionClipboardCut`, `sh::actions::common::ActionClipboardPaste`,  
`sh::actions::common::ActionClipboardPasteAsLink`, `sh::actions::common::ActionCreateFile`,  
`sh::actions::common::ActionCreateFolder`, `sh::actions::common::ActionDeleteItems`,  
`sh::actions::common::ActionExecute`, `sh::actions::common::ActionManageBookmarks`,  
`sh::actions::common::ActionNavigateInHistory`, `sh::actions::common::ActionOpenArchive`,  
`sh::actions::common::ActionOpenDirectoryInNewWindow`, `sh::actions::common::ActionOpenFile`,  
`sh::actions::common::ActionOpenFileWith::_ActionOpenFileWithEntry`, `sh::actions::common::ActionOpenFileWith::_ActionOp`,  
`sh::actions::common::ActionOpenSharcArchive`, `sh::actions::common::ActionOpenTerminalAsUser`,  
`sh::actions::common::ActionRenameItem`, `sh::actions::common::ActionShowProperties`,  
`sh::actions::common::ActionTransferToHelper`, `sh::actions::mainmenu::ActionAbout`,  
`sh::actions::mainmenu::ActionAbstractSelectNodes`, `sh::actions::mainmenu::ActionApplyProfile::ActionApplyProfileX`,  
`sh::actions::mainmenu::ActionFileViewChangeIconDimension`, `sh::actions::mainmenu::ActionFileviewFileSizeFormatting`,  
`sh::actions::mainmenu::ActionFileviewShowHiddenFiles`, `sh::actions::mainmenu::ActionFileviewView`,  
`sh::actions::mainmenu::ActionGotoDirectory`, `sh::actions::mainmenu::ActionHelp`,  
`sh::actions::mainmenu::ActionManageSavedSettings`, `sh::actions::mainmenu::ActionNumberFileviews_Add`,  
`sh::actions::mainmenu::ActionNumberFileviews_Remove`, `sh::actions::mainmenu::ActionNumberFileviews_Remove_Current`,  
`sh::actions::mainmenu::ActionQuit`, `sh::actions::mainmenu::ActionRefresh`,  
`sh::actions::mainmenu::ActionSaveSettings`, `sh::actions::mainmenu::ActionSearch`,  
`sh::actions::mainmenu::ActionSetWindowTitlePattern`, `sh::actions::mainmenu::ActionShowDetailPanel`,  
`sh::actions::mainmenu::ActionShowFolderTree`, `sh::actions::mainmenu::ActionShowLog`,  
`sh::actions::mainmenu::ActionTreeStickyness`, `sh::actions::mainmenu::ActionTuning`,  
`sh::filepropertydialogtabs::FilePropertyDialogTabExtendedAttributes::ActionAddAttribute`,  
`sh::filepropertydialogtabs::FilePropertyDialogTabExtendedAttributes::ActionChangeAttribute`,  
`sh::filepropertydialogtabs::FilePropertyDialogTabExtendedAttributes::ActionRemoveAttribute`,  
`sh::filepropertydialogtabs::FilePropertyDialogTabUnixPermissions::ActionChange`,  
`sh::filesystemhandlers::ActionMountGnomeIOLocation`, `sh::filesystemhandlers::ActionMountNetworkGnomeIOLocation`,  
`sh::filesystemhandlers::ActionUnmountGnomeIOLocation`, `sh::scripting::api::ApiActionActionItem`,  
`sh::scripting::ScriptingEngine::ActionPluginLoadCrashedAgain`, `sh::search::criteria::ActionAbstractActionDrivenSearchCriteria`

## Public Functions

**ActionActionItem** (QString *text*, bool *enabled* = true, QString *icon* = QString(), int *defaultActionPrecedence* = 0, bool *checkable* = false, bool *isChecked* = false)

Is (for subclasses) intended to be directly constructed from everywhere or by registering a factory somewhere.

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

`std::weak_ptr<AbstractActionItem> parentAction ()`  
Returns the parent action, if it is added to a container.

`void _setParentAction (std::shared_ptr<AbstractActionItem> parent)`  
Sets the parent action. .

`void initializeAsync (std::function<void>  
> oninitialized = 0)`Asynchronously initializes the action.

`void initializeSync ()`  
Synchronously initializes the action.

`bool isInitialized ()`  
Checks if this action is initialized.

`bool isInitializing ()`  
Checks if this action is initializing.

### Signals

`void changed ()`  
Emits when some data changed.

## 10.1.5 Class `sh::actions::ActionCategory`

**class** `sh::actions::ActionCategory`

Action categories are used for grouping actions in menus.

They provide the primary grouping in the toolbar and the context menus. Per default, the categories "create", "open" and "manage" exist.

### Public Functions

**ActionCategory** (QString *name*, QString *label*, QString *icon*)  
Constructed only by the infrastructure and made available otherwise.

QString **name** ()  
The category name.

QString **label** ()  
The category label.

QString **icon** ()  
The category icon.

### Public Static Functions

`std::shared_ptr<ActionCategory> getByName (QString name)`  
Gets a category by name.

`std::shared_ptr<ActionCategory> add (QString name, QString label, QString icon)`  
Adds a new category.

`QList<std::shared_ptr<ActionCategory>> categories ()`  
Returns a list of all existing categories.

`void doInitialize ()`

```
void doShutdown ()
```

### Private Members

```
QString _name
```

```
QString _label
```

```
QString _icon
```

### Private Static Attributes

```
QList<std::shared_ptr<ActionCategory>> _categories
```

## 10.1.6 Class sh::actions::ActionDefaultPrecedenceValues

```
class sh::actions::ActionDefaultPrecedenceValues
```

Reference values for calculating default precedence values of actions.

### Public Static Attributes

```
const int PRECEDENCE_OPENFILE = 300000
```

Normal file open action.

```
const int PRECEDENCE_BUILTIN_SPECIAL_OPEN = 600000
```

Special open action.

## 10.1.7 Class sh::actions::ActionExecutionInfo

```
class sh::actions::ActionExecutionInfo : public QObject
```

Programming interface for giving status infos and communication with the user in an action execution.

An instance is always available during an execution.

### Public Functions

```
ActionExecutionInfo (QObject *parent = 0)
```

Constructed only by the infrastructure and made available otherwise.

```
~ActionExecutionInfo ()
```

```
void setVisualProcessFeedbackActive (bool active)
```

Specifies if there must be a visual process feedback.

The visual process feedback is typically realized as a dialog showing progress information.

Note: It can also appear in some situations, even if this property is set to false.

```
bool isVisualProcessFeedbackActive ()
```

Determines if there must be a visual process feedback.

```
void setManualInterventionNeeded (bool isneeded)
```

Specifies if manual input is currently required.

bool **isManualInterventionNeeded** ()  
Determines if manual input is currently required.

void **startExecution** (std::shared\_ptr<sh::actions::AbstractActionItem> action)  
Triggers the beginning of the actual execution. .

void **finishExecution** ()  
Triggers the finishing of the execution. .

void **setDetails** (QString fromverb, QString fromobj, QString toverb = QString(), QString toobj =  
QString())  
Sets some detail texts.

QString **fromVerb** ()  
Gets detail text 'from-verb'.

QString **fromObjectName** ()  
Gets detail text 'from-objectname'.

QString **toVerb** ()  
Gets detail text 'to-verb'.

QString **toObjectName** ()  
Gets detail text 'to-objectname'.

void **setHead** (QString text)  
Sets the header text.

QString **head** ()  
Gets the header text.

void **setProgressIndeterminate** (QString text = QString())  
Sets progress information to 'indeterminate'.

void **setProgress** (quint64 done, quint64 all, QString text)  
Sets progress information.

bool **isProgressDeterminate** ()  
Gets if progress information is 'indeterminate'.

quint64 **progressDone** ()  
Gets count of finished items.

quint64 **progressAll** ()  
Gets count of all items.

QString **progressText** ()  
Gets progress information text.

sh::actions::ActionExecutionUserFeedback \***userfeedback** ()  
Program interface for interactive user input.

bool **isCancelled** ()  
Gets if the current action execution was cancelled.

void **cancel** ()  
Cancel the current action execution.  
  
Used from outside, e.g. as handler for a 'Cancel' button.

void **respectCancel** ()  
Called regularly from an action execution code in order to cancel execution at that place, if the user has cancelled it.



void **withExecuteGuards\_infodlg** (std::function<void>  
 >int *flags* = 0) Executes code with different exception handlers for presenting exceptions in the action execution inside the action dialog (instead of the default exception dialog).

*sh::filesystem::Operation* \***operation** ()  
 Program interface for operations on the filesystem (reading or writing).

void **addChangedEurl** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*)  
 Reminds an important change of a certain node in the filesystem (file created, modified, removed?).  
 The infrastructure will force the ui to refresh those nodes.

void **shutdown** (bool *success*)  
 Shuts down this execution info and releases resources.

## Signals

void **detailsChanged** ()  
 Signal for changed execution details.

void **headChanged** ()  
 Signal for changed head.

void **progressChanged** ()  
 Signal for changed progress.

void **visualProcessFeedbackActiveChanged** ()  
 Signal for changed visibility-required.

void **wasCancelled** ()  
 Signal for cancellation on user behalf.

bool **manualInterventionNeededChanged** ()  
 Signal for changed manual-intervention-needed.

## Private Functions

std::shared\_ptr<*sh::ui::ActionExecutionInfoDialog*> **createDialog** ()

std::shared\_ptr<*ui::ActionExecutionInfoPanel*> **createPanel** (*sh::ui::ActionExecutionInfoDialog* \**dialog*)

## Private Members

QMutex **mutex**

QString **\_head**

QString **\_fromverb**

QString **\_fromobj**

QString **\_toverb**

QString **\_toobj**

bool **\_progressDeterminate**

quint64 **\_progressDone**

quint64 **\_progressAll**

```
bool _visualProcessFeedbackActive = false
bool _wasAlwaysInvisible = true
bool _manualInterventionNeeded = false
QString _progressText
std::shared_ptr<sh::ui::ActionExecutionInfoDialog> infodialog = 0
std::shared_ptr<sh::ui::ActionExecutionInfoPanel> infopanel = 0
bool _iscancelled = false
sh::filesystem::Operation _operation
QList<std::shared_ptr<const sh::filesystem::Eurl>> changes
QStack<std::shared_ptr<sh::actions::AbstractActionItem>> stack
```

### Private Slots

```
void _checkVisibilityOnce ()
```

## 10.1.8 Class sh::actions::ActionExecutionUserFeedback

**class** *sh::actions::ActionExecutionUserFeedback*

Methods for user interaction in an action execution.

When an action executes, it may need to ask the user for some input at some time. An action implementation can do so by calling those methods. It is available as a member of the *ActionExecutionInfo* class. An instance of it is available in each action execution.

Subclassed by *sh::ui::ActionExecutionInfoDialog*

### Public Types

**enum** *MessageBoxButton*

Buttons in a message box from *ActionExecutionUserFeedback*.

*Values:*

```
enumerator NONE = 0
enumerator OK = 1 << 0
enumerator Continue = 1 << 1
enumerator Cancel = 1 << 2
enumerator Retry = 1 << 3
enumerator Yes = 1 << 4
enumerator No = 1 << 5
```

## Public Functions

```

int messageBox (QString text, QList<QString> answers, QString icon = QString(), int defaultanswer =
    -1, int cancelanswer = -1, QList<QString> answericons = QList<QString>()) = 0

int inputBox (QString text, QList<QString> answers, QString *value, QString icon = QString(), int de-
    faultanswer = -1, int cancelanswer = -1, int valuePreselectFrom = -1, int valuePreselectTo
    = -1) = 0

int multilineInputBox (QString text, QList<QString> answers, QString *value, QString icon =
    QString(), int defaultanswer = -1, int cancelanswer = -1) = 0

int simpleChooserGridform (QString text, GridformEntries *entries, QStringList answers, int de-
    faultanswer = -1, int cancelanswer = -1) = 0

bool credentialsDialog (QString text, bool showDomain, bool showUsername, bool showPass-
    word, bool showAnonymous, bool showRemember, QString *domain,
    QString *username, QString *password, bool *isAnonymous, bool *is-
    Remember) = 0

bool unixPermissionsDialog (bool *userMayRead, bool *userMayWrite, bool *userMayExecute,
    bool *groupMayRead, bool *groupMayWrite, bool *groupMayEx-
    ecute, bool *othersMayRead, bool *othersMayWrite, bool *others-
    MayExecute, bool *sticky, bool *setuid, bool *setgid, QStringList
    users, QStringList groups, QString *ownerUser, QString *owner-
    Group) = 0

QString simpleInputBox (QString text, QString deft, int valuePreselectFrom = -1, int valuePrese-
    lectTo = -1)

int simpleMessageBox (QString text, int buttons = (int)MessageBoxButton::OK, QString icon =
    QString(), int defaultbutton = (MessageBoxButton)0, int cancelbutton =
    (MessageBoxButton)0)

~ActionExecutionUserFeedback ()

class Choice
    A data structure for a choice (in an GridformEntry).

```

## Public Functions

```

Choice (QString label, QString text = QString())

```

## Public Members

```

QString label

```

```

QString text

```

```

class Choices

```

```

    A data structure for a list of Choice instances.

```

### Public Members

`QList<Choice*> list`

`int selected = -1`

**class GridformEntries**

A data structure for a list of entries in a simple chooser grid form.

### Public Functions

**GridformEntries()**

Is intended to be directly constructed from everywhere.

**~GridformEntries()**

*GridformEntry* \*newEntry (QString label)

### Public Members

`QList<GridformEntry*> list`

**class GridformEntry**

A data structure for an entry in a simple chooser grid form.

### Public Functions

**GridformEntry** (QString label, *Choices* choices)

Constructed only indirectly.

**~GridformEntry()**

**int addChoice** (QString label, QString text = QString())

### Public Members

QString label

*Choices* choices

## 10.1.9 Class sh::actions::ActionExecutionUserFeedback::Choice

**class sh::actions::ActionExecutionUserFeedback::Choice**

A data structure for a choice (in an *GridformEntry*).

### Public Functions

**Choice** (QString *label*, QString *text* = QString())

### Public Members

QString **label**

QString **text**

## 10.1.10 Class sh::actions::ActionExecutionUserFeedback::Choices

**class** *sh::actions::ActionExecutionUserFeedback::Choices*

A data structure for a list of *Choice* instances.

### Public Members

QList<*Choice*\*> **list**

int **selected** = -1

## 10.1.11 Class sh::actions::ActionExecutionUserFeedback::GridformEntries

**class** *sh::actions::ActionExecutionUserFeedback::GridformEntries*

A data structure for a list of entries in a simple chooser grid form.

### Public Functions

**GridformEntries** ()

Is intended to be directly constructed from everywhere.

**~GridformEntries** ()

*GridformEntry* \***newEntry** (QString *label*)

### Public Members

QList<*GridformEntry*\*> **list**

## 10.1.12 Class sh::actions::ActionExecutionUserFeedback::GridformEntry

**class** *sh::actions::ActionExecutionUserFeedback::GridformEntry*

A data structure for an entry in a simple chooser grid form.

### Public Functions

**GridformEntry** (QString *label*, *Choices choices*)

Constructed only indirectly.

**~GridformEntry** ()

int **addChoice** (QString *label*, QString *text* = QString())

### Public Members

QString **label**

*Choices choices*

## 10.1.13 Class sh::actions::ActionFactory

template<class T>

**class** *sh::actions::ActionFactory* : public *sh::actions::AbstractActionFactory*

A typical implementation for an action factory.

See base class for more.

### Public Functions

**ActionFactory** (std::shared\_ptr<*sh::actions::ActionCategory*> *category*,  
QList<std::shared\_ptr<*Predicate*>> *predicates*)

std::shared\_ptr<*sh::actions::AbstractActionItem*> **construct** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>>  
*nodes*)

std::shared\_ptr<*ActionInstantiation*> **actionAvailable** (*ActionInstantiation* \**instantiation*)

## 10.1.14 Class sh::actions::ActionInstantiation

**class** *sh::actions::ActionInstantiation*

Holds various data around action creation.

Action creation is a process, which begins with querying all actions available for certain nodes. The action factories evaluate availabilities and store some construction information here. Many parties are involved in working with it. The created action, is stored in this object as well.

### Public Functions

**ActionInstantiation** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> *selectedNodes* =  
{}, std::shared\_ptr<*sh::filesystem::FilesystemNode*> *currentDirectory* =  
nullptr)

Constructor for a typical fresh instance, containing infos about selected items and some more.

**ActionInstantiation** (const *ActionInstantiation* &*s*) = default

Clones an existing instance.

**ActionInstantiation** (const *ActionInstantiation* &*s*, std::shared\_ptr<*sh::actions::AbstractActionItem*>  
*createdAction*)

Clones an existing instance and sets *createdAction*. This is a convenience constructor used for some special cases.

## Public Members

QList<std::shared\_ptr<*sh::filesystem::FileSystemNode*>> **selectedNodes**

The selected nodes (initially: what's selected in the active file list).

std::shared\_ptr<*sh::filesystem::FileSystemNode*> **currentDirectory**

The current directory (which is visible in the active file list).

bool **resolveLinks** = true

If links shall be resolved for evaluation and action creation.

bool **isLookupOnCurrentDirectoryLevel** = false

If this instantiation refers to a lookup for actions on 'current directory level' (instead of 'selection level').

*AbstractActionFactory* \***actionfactory** = 0

The action faction.

std::shared\_ptr<*ActionCategory*> **category** = 0

The action category.

std::shared\_ptr<*AbstractActionItem*> **createdAction** = 0

The created action.

QKeySequence **shortcut**

The requested keyboard shortcut for the action.

int **positionIndex** = 0

The position information index.

### 10.1.15 Class *sh::actions::ActionsManager*

**class** *sh::actions::ActionsManager* : public QObject, public *sh::base::Singleton*

Factory for actions which are valid for a given list of *FileSystemNode* and tools for placing actions in some gui elements.

## Public Functions

```
void getActions (std::shared_ptr<ActionInstantiation> instantiation,
                 std::function<void> QList<std::shared_ptr<ActionInstantiation>>,
                 QList<std::shared_ptr<sh::actions::ActionInstantiation>>
                 > callback, std::function<void std::shared_ptr<ActionInstantiation>> initializedcallback
                 = [] (std::shared_ptr< ActionInstantiation >){} } Asychonously gets actions for a list of
                 sh::filesystem::FileSystemNode.
```

## Parameters

- **nodes**: The list of nodes.
- **callback**: This callback is called once the list is fetched.
- **initializedcallback**: This callback is called for each result action after initialization.

```
void getDefaultAction (QList<std::shared_ptr<sh::actions::AbstractActionItem>> actionList,
                       std::function<void> std::shared_ptr<sh::actions::ActionActionItem>
                       > callback Determines the default action for a given list of actions.
```

## Parameters

- **actionList**: The list of all available actions.

- **callback**: This callback is called when the default action was found (with an already initialized action).

void **getActionForShortcut** (QList<std::shared\_ptr<*filesystem::FilesystemNode*>> *selectedNodes*,  
std::shared\_ptr<*filesystem::FilesystemNode*> *folderNode*, int *key*, Qt::KeyboardModifiers *modifiers*,  
std::function<void> std::shared\_ptr<*sh::actions::AbstractActionItem*>  
> *callback*) Asynchronously gets the action for a shortcut.

#### Parameters

- **selectedNodes**: The nodes which are currently selected in the active fileview.
- **folderNode**: The folder which is shown by the active fileview.
- **key**: The keypress character code.
- **modifiers**: The keypress modifiers.
- **callback**: This callback is called when an action was found (with an already initialized action).

void **registerActionFactory** (std::shared\_ptr<*sh::actions::AbstractActionFactory*> *actionFactory*)

Registers an action factory.

This makes an action implementation available in the toolbar and context menus.

QList<std::shared\_ptr<*sh::actions::ActionActionItem*>> **runningActions** ()

Returns a list of all currently running actions.

void **doInitialize** ()

Executes singleton initialization.

void **doShutdown** ()

Executes singleton shutdown.

void **shutdown** ()

Shutdown down this singleton.

bool **isAlive** ()

Returns if this singleton is alive (true until its shutdown begins).

#### Signals

void **runningActionsChanged** ()

Emitted when the list of running actions changed.

#### Public Static Functions

QString **selectionLabel** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> *selection*)

Returns a short header label text for a node selection.

QString **directoryLabel** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> *directory*)

Returns a short header label text for a current directory.

void **sortAndSeparateActions** (QList<std::shared\_ptr<*ActionInstantiation*>> \**actions*)

Sorts a lists of actionitems and adds separators between groups. Modifies the given list.



## Private Functions

**ActionsManager** ()

QList<std::shared\_ptr<*ActionInstantiation*>> **\_getActions\_raw** (std::shared\_ptr<*ActionInstantiation*>  
*instantiation*)

void **\_emit\_runningActionsChanged** ()

void **recursivelyInitializeAction** (std::shared\_ptr<*sh::actions::ActionInstantiation*> *actsit*,  
std::function<void> std::shared\_ptr<*sh::actions::ActionInstantiation*>  
> *callback*, QList<std::shared\_ptr<*sh::actions::ActionInstantiation*>> *sits* = { }) Recursively initializes a list  
of action items and calls a callback for each action item recursively afterwards.

### Parameters

- *actions*: List of all action items to traverse.
- *callback*: This callback is called for each action item (even indirectly in a subtree hierarchy).

void **\_getDefaultAction\_helper** (QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> *ac-*  
*tionList*, std::shared\_ptr<*GDAHstruct*> *gdah*)  
Internal helper for the public *getDefaultAction()* method.

std::shared\_ptr<*sh::actions::ActionsManager::RunningActionsCounter*> **\_actionExecutionStarted** (std::shared\_ptr<*sh::a-*  
*ac-*  
*tion*)  
Helper for updating *\_runningactions*. .

## Private Members

QList<std::shared\_ptr<*sh::actions::AbstractActionFactory*>> **\_actionfactories**

QMutex **\_mutex\_actionfactories**

QList<std::shared\_ptr<*sh::actions::ActionActionItem*>> **\_runningactions**

QMutex **\_mutex\_runningactions**

## Friends

**friend class** ActionActionItem

**struct** *GDAHstruct*

Internal data structure used for *\_getDefaultAction\_helper()*;

## Public Functions

*GDAHstruct* () = default

*GDAHstruct* (const *GDAHstruct* &*o*) = default

### Public Members

```
sh::tools::AtomicCounter counter
std::function<void (std::shared_ptr<sh::actions::ActionActionItem>) > callback
int currentValue = 0
QList<std::shared_ptr<sh::actions::AbstractActionItem>> initialActionList
std::shared_ptr<sh::actions::AbstractActionItem> currentAction = 0

class RunningActionsCounter
    Helper for updating _runningactions. .
```

### Public Functions

```
RunningActionsCounter (ActionsManager *manager, std::shared_ptr<sh::actions::ActionActionItem>
                        action)
~RunningActionsCounter ()
```

### Private Members

```
ActionsManager *_manager
std::shared_ptr<sh::actions::ActionActionItem> _action
```

## 10.1.16 Class *sh::actions::ActionsManager::RunningActionsCounter*

```
class sh::actions::ActionsManager::RunningActionsCounter
    Helper for updating _runningactions. .
```

### Public Functions

```
RunningActionsCounter (ActionsManager *manager, std::shared_ptr<sh::actions::ActionActionItem>
                        action)
~RunningActionsCounter ()
```

### Private Members

```
ActionsManager *_manager
std::shared_ptr<sh::actions::ActionActionItem> _action
```

### 10.1.17 Class `sh::actions::ByRegExpPredicate`

**class** `sh::actions::ByRegExpPredicate` : **public** `sh::actions::Predicate`  
Shows actions only when the selection paths matches a regular expression.

#### Public Functions

**ByRegExpPredicate** (QString *regex*)

bool **evaluate** (*ActionInstantiation* \**instantiation*)  
Evaluates if the predicate is solved.

If it returns `true`, the evaluation proceeds and eventually succeeds. If `false`, it stops at this place.

void **prepare** (*ActionInstantiation* \**instantiation*)  
Prepares the evaluation.

#### Private Members

QRegExp **filterRegExp**

### 10.1.18 Class `sh::actions::DontResolveLinksPredicate`

**class** `sh::actions::DontResolveLinksPredicate` : **public** `sh::actions::Predicate`  
Disables links resolving.

#### Public Functions

**DontResolveLinksPredicate** ()

void **prepare** (*ActionInstantiation* \**instantiation*)  
Prepares the evaluation.

bool **evaluate** (*ActionInstantiation* \**instantiation*)  
Evaluates if the predicate is solved.

If it returns `true`, the evaluation proceeds and eventually succeeds. If `false`, it stops at this place.

### 10.1.19 Class `sh::actions::HeaderActionItem`

**class** `sh::actions::HeaderActionItem` : **public** `sh::actions::AbstractActionItem`  
A header.

Use it as all the other `AbstractActionItem` classes. It will lead to a visual header in the parent menu.

## Public Functions

**HeaderActionItem** (QString *text*, QString *icon* = QString())

Is intended to be directly constructed from everywhere.

QString **text** ()

Returns the displayed text for this action.

QString **icon** ()

Returns the icon for this action.

bool **enabled** ()

Checks if this action is enabled.

bool **isChecked** ()

Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>)

> *oninitialized* = 0 Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

### 10.1.20 Class `sh::actions::HideOnCurrentDirectoryLevelPredicate`

**class** `sh::actions::HideOnCurrentDirectoryLevelPredicate` : **public** `sh::actions::Predicate`  
Shows actions only when not searching for ‘current directory level’ actions.

#### Public Functions

**HideOnCurrentDirectoryLevelPredicate** ()

bool **evaluate** (*ActionInstantiation \*instantiation*)  
Evaluates if the predicate is solved.

If it returns `true`, the evaluation proceeds and eventually succeeds. If `false`, it stops at this place.

void **prepare** (*ActionInstantiation \*instantiation*)  
Prepares the evaluation.

### 10.1.21 Class `sh::actions::HideOnSelectionLevelPredicate`

**class** `sh::actions::HideOnSelectionLevelPredicate` : **public** `sh::actions::Predicate`  
Shows actions only when not searching for ‘selection level’ actions.

#### Public Functions

**HideOnSelectionLevelPredicate** ()

bool **evaluate** (*ActionInstantiation \*instantiation*)  
Evaluates if the predicate is solved.

If it returns `true`, the evaluation proceeds and eventually succeeds. If `false`, it stops at this place.

void **prepare** (*ActionInstantiation \*instantiation*)  
Prepares the evaluation.

### 10.1.22 Class `sh::actions::KeyShortcutPredicate`

**class** `sh::actions::KeyShortcutPredicate` : **public** `sh::actions::Predicate`  
Sets a keyboard shortcut.

### Public Functions

**KeyShortcutPredicate** (*QKeySequence shortcut*, *bool triggersOnCurrentDirectoryLevel*)

void **prepare** (*ActionInstantiation \*instantiation*)

Prepares the evaluation.

bool **evaluate** (*ActionInstantiation \*instantiation*)

Evaluates if the predicate is solved.

If it returns `true`, the evaluation proceeds and eventually succeeds. If `false`, it stops at this place.

### Private Members

*QKeySequence* **shortcut**

bool **triggersOnCurrentDirectoryLevel**

## 10.1.23 Class `sh::actions::OnDirectoriesPredicate`

**class** `sh::actions::OnDirectoriesPredicate` : **public** `sh::actions::Predicate`

Shows actions only on directories.

### Public Functions

**OnDirectoriesPredicate** ()

bool **evaluate** (*ActionInstantiation \*instantiation*)

Evaluates if the predicate is solved.

If it returns `true`, the evaluation proceeds and eventually succeeds. If `false`, it stops at this place.

void **prepare** (*ActionInstantiation \*instantiation*)

Prepares the evaluation.

## 10.1.24 Class `sh::actions::OnFilesPredicate`

**class** `sh::actions::OnFilesPredicate` : **public** `sh::actions::Predicate`

Shows actions only on files.

### Public Functions

**OnFilesPredicate** ()

bool **evaluate** (*ActionInstantiation \*instantiation*)

Evaluates if the predicate is solved.

If it returns `true`, the evaluation proceeds and eventually succeeds. If `false`, it stops at this place.

void **prepare** (*ActionInstantiation \*instantiation*)

Prepares the evaluation.

### 10.1.25 Class `sh::actions::OnLinksPredicate`

**class** `sh::actions::OnLinksPredicate` : **public** `sh::actions::Predicate`  
Shows actions only on links.

#### Public Functions

**OnLinksPredicate** ()

**bool evaluate** (*ActionInstantiation \*instantiation*)  
Evaluates if the predicate is solved.

If it returns `true`, the evaluation proceeds and eventually succeeds. If `false`, it stops at this place.

**void prepare** (*ActionInstantiation \*instantiation*)  
Prepares the evaluation.

### 10.1.26 Class `sh::actions::OnSingleEntrySelectionPredicate`

**class** `sh::actions::OnSingleEntrySelectionPredicate` : **public** `sh::actions::Predicate`  
Shows actions only on single-entry selections.

#### Public Functions

**OnSingleEntrySelectionPredicate** ()

**bool evaluate** (*ActionInstantiation \*instantiation*)  
Evaluates if the predicate is solved.

If it returns `true`, the evaluation proceeds and eventually succeeds. If `false`, it stops at this place.

**void prepare** (*ActionInstantiation \*instantiation*)  
Prepares the evaluation.

### 10.1.27 Class `sh::actions::PositionIndexPredicate`

**class** `sh::actions::PositionIndexPredicate` : **public** `sh::actions::Predicate`  
Sets a positioning information index.

#### Public Functions

**PositionIndexPredicate** (int *positionIndex*)

**void prepare** (*ActionInstantiation \*instantiation*)  
Prepares the evaluation.

**bool evaluate** (*ActionInstantiation \*instantiation*)  
Evaluates if the predicate is solved.

If it returns `true`, the evaluation proceeds and eventually succeeds. If `false`, it stops at this place.

## Private Members

int **positionIndex**

### 10.1.28 Class `sh::actions::Predicate`

**class** `sh::actions::Predicate`

Controls when and how an action is created in the action factory.

Subclassed by `sh::actions::ByRegExpPredicate`, `sh::actions::DontResolveLinksPredicate`,  
`sh::actions::HideOnCurrentDirectoryLevelPredicate`, `sh::actions::HideOnSelectionLevelPredicate`,  
`sh::actions::KeyShortcutPredicate`, `sh::actions::OnDirectoriesPredicate`, `sh::actions::OnFilesPredicate`,  
`sh::actions::OnLinksPredicate`, `sh::actions::OnSingleEntrySelectionPredicate`,  
`sh::actions::PositionIndexPredicate`

## Public Functions

**Predicate** ()

void **prepare** (*ActionInstantiation* \*instantiation)

Prepares the evaluation.

bool **evaluate** (*ActionInstantiation* \*instantiation)

Evaluates if the predicate is solved.

If it returns `true`, the evaluation proceeds and eventually succeeds. If `false`, it stops at this place.

**~Predicate** ()

### 10.1.29 Class `sh::actions::SeparatorActionItem`

**class** `sh::actions::SeparatorActionItem` : **public** `sh::actions::AbstractActionItem`

A separator.

Use it as all the other `AbstractActionItem` classes. It will lead to a visual separator in the parent menu.

## Public Functions

**SeparatorActionItem** ()

Is intended to be directly constructed from everywhere.

QString **text** ()

Returns the displayed text for this action.

QString **icon** ()

Returns the icon for this action.

bool **enabled** ()

Checks if this action is enabled.

bool **isChecked** ()

Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()

Checks if this action is checkable (can have a cross in the ui).



int **defaultActionPrecedence** () const

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>)

> *oninitialized* = 0 Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **changed** ()

Emits when some data changed.

### 10.1.30 Class `sh::actions::SubmenuItem`

**class** `sh::actions::SubmenuItem` : **public** `sh::actions::AbstractActionItem`

Abstract base class for a submenu.

It can be made visible in menus or the toolbar or executed directly.

Subclasses fill the submenu with other actions in order to be useful.

Subclassed by `sh::actions::common::ActionAbstractTransferTo`, `sh::actions::common::ActionBookmarkFolder`,  
`sh::actions::common::ActionGroups`, `sh::actions::common::ActionHistoryNavigate`,  
`sh::actions::common::ActionOpenFileWith`, `sh::actions::common::ActionOpenTerminal`,

*sh::actions::mainmenu::ActionApplyProfile,* *sh::actions::mainmenu::ActionMain,*  
*sh::actions::mainmenu::ActionNumberFileviews,* *sh::scripting::api::ApiSubmenuItem*

## Public Functions

**SubmenuItem** (QString *text*, **const** QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>>  
                  *subitems* = {}, bool *enabled* = true, QString *icon* = QString(), int *defaultAc-*  
                  *tionPrecedence* = 0, bool *checkable* = false, bool *ischecked* = false)

Is (for subclasses) intended to be directly constructed from everywhere or by registering a factory somewhere.

**const** QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> **subitems** ()

Returns the list of subitems from this submenu.

void **setSubitems** (**const** QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> *subitems*)

Sets the subitems.

QString **text** ()

Returns the displayed text for this action.

QString **icon** ()

Returns the icon for this action.

bool **enabled** ()

Checks if this action is enabled.

bool **isChecked** ()

Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<*AbstractActionItem*> **parentAction** ()

Returns the parent action, if it is added to a container.

```

void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.

```

## Signals

```

void subitemsChanged ()
    Emits when the list of subitems changed.

void changed ()
    Emits when some data changed.

```

## Private Members

```

QList<std::shared_ptr<sh::actions::AbstractActionItem>> _subitems

```

### 10.1.31 Class *sh::actions::common::ActionAbstractTransferTo*

**class** *sh::actions::common::ActionAbstractTransferTo* : **public** *sh::actions::SubmenuItem*  
 Abstract action for transferring a selection to some other locations (e.g. to other file views, bookmarks).  
 Subclassed by *sh::actions::common::ActionTransferToCopy*, *sh::actions::common::ActionTransferToMove*

## Public Functions

```

ActionAbstractTransferTo (QString text, QString icon, QList<std::shared_ptr<sh::filesystem::FilesystemNode>>
    nodes)

const QList<std::shared_ptr<sh::actions::AbstractActionItem>> subitems ()
    Returns the list of subitems from this submenu.

void setSubitems (const QList<std::shared_ptr<sh::actions::AbstractActionItem>> subitems)
    Sets the subitems.

QString text ()
    Returns the displayed text for this action.

QString icon ()
    Returns the icon for this action.

bool enabled ()
    Checks if this action is enabled.

bool isChecked ()
    Checks if this action is checked (has a cross in the ui).

```

bool **isCheckedable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>

> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **subitemsChanged** ()

Emits when the list of subitems changed.

void **changed** ()

Emits when some data changed.

## Friends

**friend class** ActionTransferToHelper

### 10.1.32 Class `sh::actions::common::ActionAbstractTransferTree`

**class** `sh::actions::common::ActionAbstractTransferTree` : **public** `sh::actions::ActionActionItem`

Abstract action for transferring (copying, moving, et al) a tree of items.

Subclassed by `sh::actions::common::ActionCopyTree`

## Public Functions

**ActionAbstractTransferTree** (QList<std::shared\_ptr<const `sh::filesystem::Eurl`>> *sources*,  
std::shared\_ptr<const `sh::filesystem::Eurl`> *destination*)

**~ActionAbstractTransferTree** ()

void **execute** ()

Executes this action.

void **execute** (`sh::actions::ActionExecutionInfo` \**info*)

Executes this action.

QKeySequence **shortcuthint** ()

Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()

Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)

Sets the keyboard shortcut for triggering this action.

QString **text** ()

Returns the displayed text for this action.

QString **icon** ()

Returns the icon for this action.

bool **enabled** ()

Checks if this action is enabled.

bool **isChecked** ()

Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<AbstractActionItem> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_\_setParentAction** (std::shared\_ptr<AbstractActionItem> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

## Public Static Functions

void **makereadableunits** (double *\*value*, QString *\*vunit*)

**class MyFilesystemOperationProgressMonitor** : public *sh::filesystem::FilesystemOperationProgressMonitor*

## Public Functions

**MyFilesystemOperationProgressMonitor** (*ActionExecutionInfo \*info*)

bool **hasItemInfo** ()  
Checks if this progress monitor provides information about how many items of a certain total number are transferred so far.

quint64 **doneItems** ()  
Checks how many items are transferred so far.

quint64 **allItems** ()  
Checks how many items are to be transferred in total (as predicted in the current moment).

**bool hasBytesInfo ()**  
 Checks if this progress monitor provides information about how many byte of a certain total number are transferred so far.

**quint64 doneBytes ()**  
 Checks how many bytes are transferred so far.

**quint64 allBytes ()**  
 Checks how many bytes are to be transferred in total (as predicted in the current moment).

**QString getItemInfoFrom ()**  
 Returns the current source of transfer (as textual information).

**QString getItemInfoTo ()**  
 Returns the current destination of transfer (as textual information).

**QString estimation ()**  
 Returns the current performance and time estimation (as textual information).

### 10.1.33 Class `sh::actions::common::ActionAbstractTransferTree::MyFilesystemOperationProgressMonitor`

`class sh::actions::common::ActionAbstractTransferTree::MyFilesystemOperationProgressMonitor`

#### Public Functions

**MyFilesystemOperationProgressMonitor** (*ActionExecutionInfo \*info*)

**bool hasItemInfo ()**  
 Checks if this progress monitor provides information about how many items of a certain total number are transferred so far.

**quint64 doneItems ()**  
 Checks how many items are transferred so far.

**quint64 allItems ()**  
 Checks how many items are to be transferred in total (as predicted in the current moment).

**bool hasBytesInfo ()**  
 Checks if this progress monitor provides information about how many byte of a certain total number are transferred so far.

**quint64 doneBytes ()**  
 Checks how many bytes are transferred so far.

**quint64 allBytes ()**  
 Checks how many bytes are to be transferred in total (as predicted in the current moment).

**QString getItemInfoFrom ()**  
 Returns the current source of transfer (as textual information).

**QString getItemInfoTo ()**  
 Returns the current destination of transfer (as textual information).

**QString estimation ()**  
 Returns the current performance and time estimation (as textual information).

### 10.1.34 Class `sh::actions::common::ActionAddToBookmarks`

**class** `sh::actions::common::ActionAddToBookmarks` : **public** `sh::actions::ActionActionItem`  
Action for bookmarking a directory.

#### Public Functions

**ActionAddToBookmarks** (`QList<std::shared_ptr<sh::filesystem::FilesystemNode>> nodes`)

**void execute** ()  
Executes this action.

**void execute** (*sh::actions::ActionExecutionInfo* \*info)  
Executes this action.

**QKeySequence shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

**bool shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

**void setShortcuthint** (`QKeySequence shortcut`, `bool triggersOnCurrentDirectory = false`)  
Sets the keyboard shortcut for triggering this action.

**QString text** ()  
Returns the displayed text for this action.

**QString icon** ()  
Returns the icon for this action.

**bool enabled** ()  
Checks if this action is enabled.

**bool isChecked** ()  
Checks if this action is checked (has a cross in the ui).

**bool isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

**int defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

**void setText** (`QString text`)  
Sets the displayed text.

**void setIcon** (`QString icon`)  
Sets the icon.

**void setEnabled** (`bool enabled`)  
Sets if the item is enabled.

**void setChecked** (`bool checked`)  
Sets if the item is checked (has a cross in the ui).

**void setVisible** (`bool visible`)  
Sets the visibility of this item.



```

bool visible ()
    Checks the visibility of this item (non-recursively).

std::weak_ptr<AbstractActionItem> parentAction ()
    Returns the parent action, if it is added to a container.

void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.

```

## Signals

```

void changed ()
    Emits when some data changed.

```

## Public Static Functions

```

void doInitialize ()

void doShutdown ()

```

### 10.1.35 Class `sh::actions::common::ActionBookmark`

```

class sh::actions::common::ActionBookmark : public sh::actions::ActionActionItem
    Action for navigating to a certain bookmark.

```

## Public Functions

```

ActionBookmark (QString label, std::shared_ptr<const sh::filesystem::Eurl> eurl)

void execute ()
    Executes this action.

void execute (sh::actions::ActionExecutionInfo *info)
    Executes this action.

QKeySequence shortcuthint ()
    Returns the keyboard shortcut for triggering this action.

bool shortcuthintTriggersOnCurrentDirectory ()
    Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry
    selection).

void setShortcuthint (QKeySequence shortcut, bool triggersOnCurrentDirectory = false)
    Sets the keyboard shortcut for triggering this action.

```

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

### 10.1.36 Class `sh::actions::common::ActionBookmarkFolder`

**class** `sh::actions::common::ActionBookmarkFolder` : **public** `sh::actions::SubmenuItem`  
Submenu action of bookmarks.

Subclassed by `sh::actions::common::ActionBookmarks`

#### Public Functions

**ActionBookmarkFolder** (QString *label*)

void **addSubitem** (std::shared\_ptr<`sh::actions::AbstractActionItem`> *item*)

void **clearSubitems** ()

**const** QList<std::shared\_ptr<`sh::actions::AbstractActionItem`>> **subitems** ()  
Returns the list of subitems from this submenu.

void **setSubitems** (**const** QList<std::shared\_ptr<`sh::actions::AbstractActionItem`>> *subitems*)  
Sets the subitems.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0)  
Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **subitemsChanged** ()  
Emits when the list of subitems changed.

void **changed** ()  
Emits when some data changed.

### 10.1.37 Class `sh::actions::common::ActionBookmarks`

**class** `sh::actions::common::ActionBookmarks` : **public** `sh::actions::common::ActionBookmarkFolder`  
Bookmarking action (the main one you see in the toolbar)

#### Public Functions

**ActionBookmarks** ()

void **initialize** ()  
Initialize the action. This should make the time-consuming parts, e.g. for determining a label or enabled state.

void **addSubitem** (std::shared\_ptr<[sh::actions::AbstractActionItem](#)> *item*)

void **clearSubitems** ()

**const** QList<std::shared\_ptr<[sh::actions::AbstractActionItem](#)>> **subitems** ()  
Returns the list of subitems from this submenu.

void **setSubitems** (**const** QList<std::shared\_ptr<[sh::actions::AbstractActionItem](#)>> *subitems*)  
Sets the subitems.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **subitemsChanged** ()  
Emits when the list of subitems changed.

void **changed** ()  
Emits when some data changed.

### 10.1.38 Class `sh::actions::common::ActionClipboardCopy`

**class** `sh::actions::common::ActionClipboardCopy` : **public** `sh::actions::ActionActionItem`  
Action for copying to clipboard.

#### Public Functions

**ActionClipboardCopy** (QList<std::shared\_ptr<`sh::filesystem::FilesystemNode`>> *nodes*)

void **execute** ()  
Executes this action.

void **execute** (`sh::actions::ActionExecutionInfo` \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

```

void setIcon (QString icon)
    Sets the icon.

void setEnabled (bool enabled)
    Sets if the item is enabled.

void setChecked (bool checked)
    Sets if the item is checked (has a cross in the ui).

void setVisible (bool visible)
    Sets the visibility of this item.

bool visible ()
    Checks the visibility of this item (non-recursively).

std::weak_ptr<AbstractActionItem> parentAction ()
    Returns the parent action, if it is added to a container.

void __setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.

```

## Signals

```

void changed ()
    Emits when some data changed.

```

## Public Static Functions

```

void doInitialize ()

void doShutdown ()

```

### 10.1.39 Class sh::actions::common::ActionClipboardCut

```

class sh::actions::common::ActionClipboardCut : public sh::actions::ActionActionItem
    Action for cutting to clipboard.

```

## Public Functions

**ActionClipboardCut** (QList<std::shared\_ptr<[sh::filesystem::FilesystemNode](#)>> *nodes*)

void **execute** ()  
Executes this action.

void **execute** ([sh::actions::ActionExecutionInfo](#) \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.



```
void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.
```

## Signals

```
void changed ()
    Emits when some data changed.
```

## Public Static Functions

```
void doInitialize ()

void doShutdown ()
```

### 10.1.40 Class `sh::actions::common::ActionClipboardPaste`

```
class sh::actions::common::ActionClipboardPaste : public sh::actions::ActionActionItem
    Action for pasting from clipboard.
```

## Public Functions

```
ActionClipboardPaste (QList<std::shared_ptr<sh::filesystem::FilesystemNode>> nodes)

bool shortcutTriggersOnFolder ()

void execute ()
    Executes this action.

void execute (sh::actions::ActionExecutionInfo *info)
    Executes this action.

QKeySequence shortcuthint ()
    Returns the keyboard shortcut for triggering this action.

bool shortcuthintTriggersOnCurrentDirectory ()
    Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry
    selection).

void setShortcuthint (QKeySequence shortcut, bool triggersOnCurrentDirectory = false)
    Sets the keyboard shortcut for triggering this action.

QString text ()
    Returns the displayed text for this action.

QString icon ()
    Returns the icon for this action.
```

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

## Public Static Functions

void **doInitialize** ()  
void **doShutdown** ()

### 10.1.41 Class `sh::actions::common::ActionClipboardPasteAsLink`

**class** `sh::actions::common::ActionClipboardPasteAsLink` : **public** `sh::actions::ActionActionItem`  
Action for pasting from clipboard as link.

## Public Functions

**ActionClipboardPasteAsLink** (QList<std::shared\_ptr<`sh::filesystem::FilesystemNode`>> *nodes*)

void **execute** ()  
Executes this action.

void **execute** (`sh::actions::ActionExecutionInfo` \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<AbstractActionItem> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<AbstractActionItem> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

## Public Static Functions

void **doInitialize** ()  
void **doShutdown** ()

### 10.1.42 Class sh::actions::common::ActionCopyTree

**class** *sh::actions::common::ActionCopyTree* : **public** *sh::actions::common::ActionAbstractTransferTree*  
Action copying a filesystem tree to some other place.  
Subclassed by *sh::actions::common::ActionMoveTree*

## Public Functions

**ActionCopyTree** (QList<std::shared\_ptr<const *sh::filesystem::Eurl*>> *sources*,  
std::shared\_ptr<const *sh::filesystem::Eurl*> *destination*)

void **action** (*sh::actions::ActionExecutionInfo* \**info*)  
The action implementation, i.e. what the actions should actually do.

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

`std::weak_ptr<AbstractActionItem> parentAction ()`  
Returns the parent action, if it is added to a container.

`void _setParentAction (std::shared_ptr<AbstractActionItem> parent)`  
Sets the parent action. .

`void initializeAsync (std::function<void>  
> oninitialized = 0)`Asynchronously initializes the action.

`void initializeSync ()`  
Synchronously initializes the action.

`bool isInitialized ()`  
Checks if this action is initialized.

`bool isInitializing ()`  
Checks if this action is initializing.

## Signals

`void changed ()`  
Emits when some data changed.

## Public Static Functions

`void makereadableunits (double *value, QString *vunit)`

### 10.1.43 Class `sh::actions::common::ActionCreateFile`

`class sh::actions::common::ActionCreateFile : public sh::actions::ActionActionItem`  
Action for creating a new file.

## Public Functions

`ActionCreateFile (QList<std::shared_ptr<sh::filesystem::FilesystemNode>> nodes)`

`void execute ()`  
Executes this action.

`void execute (sh::actions::ActionExecutionInfo *info)`  
Executes this action.

`QKeySequence shortcuthint ()`  
Returns the keyboard shortcut for triggering this action.

`bool shortcuthintTriggersOnCurrentDirectory ()`  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

`void setShortcuthint (QKeySequence shortcut, bool triggersOnCurrentDirectory = false)`  
Sets the keyboard shortcut for triggering this action.

`QString text ()`  
Returns the displayed text for this action.

`QString icon ()`  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

## Public Static Functions

void **doInitialize** ()  
void **doShutdown** ()

### 10.1.44 Class `sh::actions::common::ActionCreateFolder`

**class** `sh::actions::common::ActionCreateFolder` : **public** `sh::actions::ActionActionItem`  
Action for creating a new directory.

## Public Functions

**ActionCreateFolder** (QList<std::shared\_ptr<`sh::filesystem::FilesystemNode`>> *nodes*)  
**bool shortcutTriggersOnFolder** ()  
void **execute** ()  
Executes this action.  
void **execute** (`sh::actions::ActionExecutionInfo` \**info*)  
Executes this action.  
QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.  
**bool shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).  
void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.  
QString **text** ()  
Returns the displayed text for this action.  
QString **icon** ()  
Returns the icon for this action.  
**bool enabled** ()  
Checks if this action is enabled.  
**bool isChecked** ()  
Checks if this action is checked (has a cross in the ui).  
**bool isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).  
**int defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.



The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

```
void setText (QString text)
    Sets the displayed text.

void setIcon (QString icon)
    Sets the icon.

void setEnabled (bool enabled)
    Sets if the item is enabled.

void setChecked (bool checked)
    Sets if the item is checked (has a cross in the ui).

void setVisible (bool visible)
    Sets the visibility of this item.

bool visible ()
    Checks the visibility of this item (non-recursively).

std::weak_ptr<AbstractActionItem> parentAction ()
    Returns the parent action, if it is added to a container.

void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.
```

## Signals

```
void changed ()
    Emits when some data changed.
```

## Public Static Functions

```
void doInitialize ()

void doShutdown ()
```

### 10.1.45 Class `sh::actions::common::ActionDeleteItems`

**class** `sh::actions::common::ActionDeleteItems` : **public** `sh::actions::ActionActionItem`  
Action deleting a list of filesystem items.

#### Public Functions

**ActionDeleteItems** (`QList<std::shared_ptr<sh::filesystem::FilesystemNode>> nodes`)

**void execute** ()  
Executes this action.

**void execute** (`sh::actions::ActionExecutionInfo *info`)  
Executes this action.

**QKeySequence shortcutHint** ()  
Returns the keyboard shortcut for triggering this action.

**bool shortcutHintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

**void setShortcutHint** (`QKeySequence shortcut`, `bool triggersOnCurrentDirectory = false`)  
Sets the keyboard shortcut for triggering this action.

**QString text** ()  
Returns the displayed text for this action.

**QString icon** ()  
Returns the icon for this action.

**bool enabled** ()  
Checks if this action is enabled.

**bool isChecked** ()  
Checks if this action is checked (has a cross in the ui).

**bool isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

**int defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

**void setText** (`QString text`)  
Sets the displayed text.

**void setIcon** (`QString icon`)  
Sets the icon.

**void setEnabled** (`bool enabled`)  
Sets if the item is enabled.

**void setChecked** (`bool checked`)  
Sets if the item is checked (has a cross in the ui).

**void setVisible** (`bool visible`)  
Sets the visibility of this item.

bool **visible** ()  
 Checks the visibility of this item (non-recursively).

std::weak\_ptr<*AbstractActionItem*> **parentAction** ()  
 Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<*AbstractActionItem*> *parent*)  
 Sets the parent action. .

void **initializeAsync** (std::function<void>  
 > *oninitialized* = 0)  
 Asynchronously initializes the action.

void **initializeSync** ()  
 Synchronously initializes the action.

bool **isInitialized** ()  
 Checks if this action is initialized.

bool **isInitializing** ()  
 Checks if this action is initializing.

## Signals

void **changed** ()  
 Emits when some data changed.

## Public Static Functions

void **doInitialize** ()  
 void **doShutdown** ()

### 10.1.46 Class sh::actions::common::ActionExecute

**class** *sh::actions::common::ActionExecute* : **public** *sh::actions::ActionActionItem*  
 Action executing a file (if executable).

## Public Functions

**ActionExecute** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> *nodes*)

void **execute** ()  
 Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)  
 Executes this action.

QKeySequence **shortcuthint** ()  
 Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
 Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
 Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

## Public Static Functions

void **doInitialize** ()  
void **doShutdown** ()

### 10.1.47 Class `sh::actions::common::ActionGroups`

**class** `sh::actions::common::ActionGroups` : **public** `sh::actions::SubmenuItem`  
Action for showing registered actions grouped by an `sh::actions::ActionCategory`.

## Public Functions

**ActionGroups** (std::shared\_ptr<`sh::actions::ActionCategory`> *category*)  
void **setGroupActions** (std::shared\_ptr<`sh::actions::ActionInstantiation`> *rootsituation*,  
QList<std::shared\_ptr<`sh::actions::ActionInstantiation`>> *selacts*,  
QList<std::shared\_ptr<`sh::actions::ActionInstantiation`>> *diracts*)  
**const** QList<std::shared\_ptr<`sh::actions::AbstractActionItem`>> **subitems** ()  
Returns the list of subitems from this submenu.  
void **setSubitems** (**const** QList<std::shared\_ptr<`sh::actions::AbstractActionItem`>> *subitems*)  
Sets the subitems.  
QString **text** ()  
Returns the displayed text for this action.  
QString **icon** ()  
Returns the icon for this action.  
bool **enabled** ()  
Checks if this action is enabled.  
bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).  
bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).  
int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.  
void **setText** (QString *text*)  
Sets the displayed text.  
void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0)  
Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **subitemsChanged** ()  
Emits when the list of subitems changed.

void **changed** ()  
Emits when some data changed.

### 10.1.48 Class `sh::actions::common::ActionHistoryNavigate`

**class** `sh::actions::common::ActionHistoryNavigate` : **public** `sh::actions::SubmenuItem`  
Abstract action for navigating in the directory history stack (stepwise or randomly).

Subclassed by `sh::actions::common::ActionHistoryNavigateBackward`, `sh::actions::common::ActionHistoryNavigateForward`

## Public Functions

**ActionHistoryNavigate** (QString *text*, std::function<QList<`sh::tools::HistoryTracker`<std::shared\_ptr<const  
`sh::filesystem::Eurl`>>::HistoryEntry>  
> *\_getentriesfct*, QString *icon*)

**const** QList<std::shared\_ptr<`sh::actions::AbstractActionItem`>> **subitems** ()  
Returns the list of subitems from this submenu.

void **setSubitems** (**const** QList<std::shared\_ptr<`sh::actions::AbstractActionItem`>> *subitems*)  
Sets the subitems.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **subitemsChanged** ()  
Emits when the list of subitems changed.

void **changed** ()  
Emits when some data changed.

### 10.1.49 Class `sh::actions::common::ActionHistoryNavigateBackward`

**class** `sh::actions::common::ActionHistoryNavigateBackward` : **public** `sh::actions::common::ActionHistoryNav`  
Action for navigating backward in the directory history stack.

#### Public Functions

**ActionHistoryNavigateBackward** ()

**const** `QList<std::shared_ptr<sh::actions::AbstractActionItem>>` **subitems** ()  
Returns the list of subitems from this submenu.

void **setSubitems** (**const** `QList<std::shared_ptr<sh::actions::AbstractActionItem>>` *subitems*)  
Sets the subitems.

`QString` **text** ()  
Returns the displayed text for this action.

`QString` **icon** ()  
Returns the icon for this action.

**bool** **enabled** ()  
Checks if this action is enabled.

**bool** **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

**bool** **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

**int** **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (`QString` *text*)  
Sets the displayed text.

void **setIcon** (`QString` *icon*)  
Sets the icon.

void **setEnabled** (**bool** *enabled*)  
Sets if the item is enabled.

void **setChecked** (**bool** *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (**bool** *visible*)  
Sets the visibility of this item.



bool **visible** ()  
 Checks the visibility of this item (non-recursively).

std::weak\_ptr<*AbstractActionItem*> **parentAction** ()  
 Returns the parent action, if it is added to a container.

void **setParentAction** (std::shared\_ptr<*AbstractActionItem*> parent)  
 Sets the parent action. .

void **initializeAsync** (std::function<void>  
 > oninitialized = 0) Asynchronously initializes the action.

void **initializeSync** ()  
 Synchronously initializes the action.

bool **isInitialized** ()  
 Checks if this action is initialized.

bool **isInitializing** ()  
 Checks if this action is initializing.

## Signals

void **subitemsChanged** ()  
 Emits when the list of subitems changed.

void **changed** ()  
 Emits when some data changed.

### 10.1.50 Class sh::actions::common::ActionHistoryNavigateForward

**class** *sh::actions::common::ActionHistoryNavigateForward* : **public** *sh::actions::common::ActionHistoryNavigateForward*  
 Action for navigating forward in the directory history stack.

#### Public Functions

**ActionHistoryNavigateForward** ()

**const** QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> **subitems** ()  
 Returns the list of subitems from this submenu.

void **setSubitems** (**const** QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> subitems)  
 Sets the subitems.

QString **text** ()  
 Returns the displayed text for this action.

QString **icon** ()  
 Returns the icon for this action.

bool **enabled** ()  
 Checks if this action is enabled.

bool **isChecked** ()  
 Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
 Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () const

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>)

> *oninitialized* = 0 Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **subitemsChanged** ()

Emits when the list of subitems changed.

void **changed** ()

Emits when some data changed.

### 10.1.51 Class `sh::actions::common::ActionManageBookmarks`

**class** `sh::actions::common::ActionManageBookmarks` : public `sh::actions::ActionActionItem`  
 Action for managing bookmarks.

#### Public Functions

**ActionManageBookmarks** ()

void **execute** ()  
 Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo \*info*)  
 Executes this action.

QKeySequence **shortcuthint** ()  
 Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
 Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
 Sets the keyboard shortcut for triggering this action.

QString **text** ()  
 Returns the displayed text for this action.

QString **icon** ()  
 Returns the icon for this action.

bool **enabled** ()  
 Checks if this action is enabled.

bool **isChecked** ()  
 Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
 Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
 Returns the precedence for becoming the default action of a parent submenu.  
 This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
 Sets the displayed text.

void **setIcon** (QString *icon*)  
 Sets the icon.

void **setEnabled** (bool *enabled*)  
 Sets if the item is enabled.

void **setChecked** (bool *checked*)  
 Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
 Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

### 10.1.52 Class `sh::actions::common::ActionMoveTree`

**class** `sh::actions::common::ActionMoveTree` : **public** `sh::actions::common::ActionCopyTree`  
Action moving a filesystem tree to some other place.

#### Public Functions

**ActionMoveTree** (QList<std::shared\_ptr<**const** [sh::filesystem::Eurl](#)>> *sources*,  
std::shared\_ptr<**const** [sh::filesystem::Eurl](#)> *destination*)

void **action** ([sh::actions::ActionExecutionInfo](#) \**info*)  
The action implementation, i.e. what the actions should actually do.

void **execute** ()  
Executes this action.

void **execute** ([sh::actions::ActionExecutionInfo](#) \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckedable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

## Public Static Functions

void **makereadableunits** (double *\*value*, QString *\*vunit*)

### 10.1.53 Class `sh::actions::common::ActionNavigateInHistory`

**class** `sh::actions::common::ActionNavigateInHistory` : **public** `sh::actions::ActionActionItem`  
Action for navigating to one particular place in the directory history stack.

## Public Functions

**ActionNavigateInHistory** (`sh::ui::FileView` *\*filelist*, int *idx*, std::shared\_ptr<const  
*sh::filesystem::Eurl*> *eurl*, bool *isdefault*)

void **execute** ()  
Executes this action.

void **execute** (`sh::actions::ActionExecutionInfo` *\*info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

```

void setText (QString text)
    Sets the displayed text.

void setIcon (QString icon)
    Sets the icon.

void setEnabled (bool enabled)
    Sets if the item is enabled.

void setChecked (bool checked)
    Sets if the item is checked (has a cross in the ui).

void setVisible (bool visible)
    Sets the visibility of this item.

bool visible ()
    Checks the visibility of this item (non-recursively).

std::weak_ptr<AbstractActionItem> parentAction ()
    Returns the parent action, if it is added to a container.

void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.

```

## Signals

```

void changed ()
    Emits when some data changed.

```

### 10.1.54 Class `sh::actions::common::ActionOpenArchive`

```

class sh::actions::common::ActionOpenArchive : public sh::actions::ActionActionItem
    Action opening a share archive (by creating a root node for it).

    Subclassed      by      sh::filesystemhandlers::ArchiveFilesystemHandler::ActionOpenTarArchive,
    sh::filesystemhandlers::ArchiveFilesystemHandler::ActionOpenZipArchive

```

## Public Functions

**ActionOpenArchive** (QList<std::shared\_ptr<[sh::filesystem::FilesystemNode](#)>> *nodes*)

void **execute** ()  
Executes this action.

void **execute** ([sh::actions::ActionExecutionInfo](#) \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.



```
void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.
```

## Signals

```
void changed ()
    Emits when some data changed.
```

### 10.1.55 Class `sh::actions::common::ActionOpenDirectoryInNewWindow`

```
class sh::actions::common::ActionOpenDirectoryInNewWindow : public sh::actions::ActionActionItem
    Action for opening a file with a automatically chosen program.
```

## Public Functions

```
ActionOpenDirectoryInNewWindow (QList<std::shared_ptr<sh::filesystem::FilesystemNode>>
    nodes)

void execute ()
    Executes this action.

void execute (sh::actions::ActionExecutionInfo *info)
    Executes this action.

QKeySequence shortcuthint ()
    Returns the keyboard shortcut for triggering this action.

bool shortcuthintTriggersOnCurrentDirectory ()
    Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry
    selection).

void setShortcuthint (QKeySequence shortcut, bool triggersOnCurrentDirectory = false)
    Sets the keyboard shortcut for triggering this action.

QString text ()
    Returns the displayed text for this action.

QString icon ()
    Returns the icon for this action.

bool enabled ()
    Checks if this action is enabled.

bool isChecked ()
    Checks if this action is checked (has a cross in the ui).
```

bool **isCheckedable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>

> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **changed** ()

Emits when some data changed.

## Public Static Functions

void **doInitialize** ()

void **doShutdown** ()

### 10.1.56 Class `sh::actions::common::ActionOpenFile`

**class** `sh::actions::common::ActionOpenFile` : **public** `sh::actions::ActionActionItem`  
 Action for opening a file with a automatically chosen program.

## Public Functions

**ActionOpenFile** (QList<std::shared\_ptr<`sh::filesystem::FilesystemNode`>> *nodes*)

void **execute** ()  
 Executes this action.

void **execute** (`sh::actions::ActionExecutionInfo` \**info*)  
 Executes this action.

QKeySequence **shortcuthint** ()  
 Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
 Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
 Sets the keyboard shortcut for triggering this action.

QString **text** ()  
 Returns the displayed text for this action.

QString **icon** ()  
 Returns the icon for this action.

bool **enabled** ()  
 Checks if this action is enabled.

bool **isChecked** ()  
 Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
 Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
 Returns the precedence for becoming the default action of a parent submenu.  
 This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
 Sets the displayed text.

void **setIcon** (QString *icon*)  
 Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

## Public Static Functions

void **doInitialize** ()  
void **doShutdown** ()

### 10.1.57 Class `sh::actions::common::ActionOpenFileWith`

**class** `sh::actions::common::ActionOpenFileWith`: **public** `sh::actions::SubmenuItem`  
Submenu action for opening a file with a manually chosen program.

## Public Functions

**ActionOpenFileWith** (QList<std::shared\_ptr<[sh::filesystem::FilesystemNode](#)>> *nodes*)

**const** QList<std::shared\_ptr<[sh::actions::AbstractActionItem](#)>> **subitems** ()  
Returns the list of subitems from this submenu.

void **setSubitems** (**const** QList<std::shared\_ptr<[sh::actions::AbstractActionItem](#)>> *subitems*)  
Sets the subitems.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **subitemsChanged** ()  
Emits when the list of subitems changed.

void **changed** ()  
Emits when some data changed.

## Public Static Functions

void **waitProgramClosedIfNeeded** (*sh::actions::ActionExecutionInfo* \*info, QStringList *filelist*)

void **doInitialize** ()

void **doShutdown** ()

## Friends

**friend class** ActionOpenFile

**class** **\_ActionOpenFileWithEntry** : **public** *sh::actions::ActionActionItem*

## Public Functions

**\_ActionOpenFileWithEntry** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>>  
*nodes*, QString *name*, QString *cmd*, QStringList *arguments*,  
QStringList *rememberformimetype*)

void **action** (*sh::actions::ActionExecutionInfo* \*info)  
The action implementation, i.e. what the actions should actually do.

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \*info)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckedable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () const

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>)

> *oninitialized* = 0 Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Public Members

QString **\_\_command**

QStringList **\_\_commandArguments**

QList<std::shared\_ptr<[sh::filesystem::FilesystemNode](#)>> **\_\_nodes**

## Signals

void **changed** ()  
Emits when some data changed.

**class** **\_ActionOpenFileWithUI** : **public** *sh::actions::ActionActionItem*

## Public Functions

**\_ActionOpenFileWithUI** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> *nodes*,  
QString *mimetype*)

void **action** (*sh::actions::ActionExecutionInfo* \**info*)  
The action implementation, i.e. what the actions should actually do.

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.



```

void setChecked (bool checked)
    Sets if the item is checked (has a cross in the ui).

void setVisible (bool visible)
    Sets the visibility of this item.

bool visible ()
    Checks the visibility of this item (non-recursively).

std::weak_ptr<AbstractActionItem> parentAction ()
    Returns the parent action, if it is added to a container.

void __setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.

```

### Public Members

```

QList<std::shared_ptr<sh::filesystem::FileSystemNode>> _nodes

QString _mimetype

```

### Signals

```

void changed ()
    Emits when some data changed.

```

## 10.1.58 Class `sh::actions::common::ActionOpenFileWith::RememberedOpenAction`

```
class sh::actions::common::ActionOpenFileWith::RememberedOpenAction
```

### Public Functions

```

RememberedOpenAction (QString command, QStringList arguments)

bool operator== (const RememberedOpenAction &a2) const

```

## Public Members

QString **command**

QStringList **arguments**

## 10.1.59 Class `sh::actions::common::ActionOpenFileWith::_ActionOpenFileWithEntry`

```
class sh::actions::common::ActionOpenFileWith::_ActionOpenFileWithEntry : public sh::actions::ActionOpenFileWith
```

## Public Functions

**\_ActionOpenFileWithEntry** (QList<std::shared\_ptr<sh::filesystem::FilesystemNode>> *nodes*,  
QString *name*, QString *cmd*, QStringList *arguments*, QStringList  
*rememberformimetype*)

void **action** (sh::actions::ActionExecutionInfo \**info*)  
The action implementation, i.e. what the actions should actually do.

void **execute** ()  
Executes this action.

void **execute** (sh::actions::ActionExecutionInfo \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

```

void setIcon (QString icon)
    Sets the icon.

void setEnabled (bool enabled)
    Sets if the item is enabled.

void setChecked (bool checked)
    Sets if the item is checked (has a cross in the ui).

void setVisible (bool visible)
    Sets the visibility of this item.

bool visible ()
    Checks the visibility of this item (non-recursively).

std::weak_ptr<AbstractActionItem> parentAction ()
    Returns the parent action, if it is added to a container.

void setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.

```

## Public Members

```

QString _command
QStringList _commandArguments
QList<std::shared_ptr<sh::filesystem::FilesystemNode>> _nodes

```

## Signals

```

void changed ()
    Emits when some data changed.

```

### 10.1.60 Class sh::actions::common::ActionOpenFileWith::\_ActionOpenFileWithUI

```

class sh::actions::common::ActionOpenFileWith::_ActionOpenFileWithUI : public sh::actions::ActionAct

```

## Public Functions

**\_ActionOpenFileWithUI** (QList<std::shared\_ptr<[sh::filesystem::FilesystemNode](#)>> *nodes*, QString *mimetype*)

void **action** ([sh::actions::ActionExecutionInfo](#) \**info*)  
The action implementation, i.e. what the actions should actually do.

void **execute** ()  
Executes this action.

void **execute** ([sh::actions::ActionExecutionInfo](#) \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

```

std::weak_ptr<AbstractActionItem> parentAction ()
    Returns the parent action, if it is added to a container.

void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.

```

### Public Members

```

QList<std::shared_ptr<sh::filesystem::FilesystemNode>> _nodes

QString _mimetype

```

### Signals

```

void changed ()
    Emits when some data changed.

```

## 10.1.61 Class sh::actions::common::ActionOpenSharcArchive

```

class sh::actions::common::ActionOpenSharcArchive : public sh::actions::ActionActionItem
    Action opening a sharc archive (by creating a root node for it).

```

### Public Functions

```

ActionOpenSharcArchive (QList<std::shared_ptr<sh::filesystem::FilesystemNode>> nodes)

void execute ()
    Executes this action.

void execute (sh::actions::ActionExecutionInfo *info)
    Executes this action.

QKeySequence shortcuthint ()
    Returns the keyboard shortcut for triggering this action.

bool shortcuthintTriggersOnCurrentDirectory ()
    Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry
    selection).

void setShortcuthint (QKeySequence shortcut, bool triggersOnCurrentDirectory = false)
    Sets the keyboard shortcut for triggering this action.

QString text ()
    Returns the displayed text for this action.

```

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

## Public Static Functions

void **doInitialize** ()  
void **doShutdown** ()

### 10.1.62 Class `sh::actions::common::ActionOpenTerminal`

**class** `sh::actions::common::ActionOpenTerminal` : **public** `sh::actions::SubmenuItem`  
Action submenu with actions for opening commandline terminals.

## Public Functions

**ActionOpenTerminal** (QString *dir*)

void **initialize** ()  
Initialize the action. This should make the time-consuming parts, e.g. for determining a label or enabled state.

**const** QList<std::shared\_ptr<`sh::actions::AbstractActionItem`>> **subitems** ()  
Returns the list of subitems from this submenu.

void **setSubitems** (**const** QList<std::shared\_ptr<`sh::actions::AbstractActionItem`>> *subitems*)  
Sets the subitems.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **subitemsChanged** ()  
Emits when the list of subitems changed.

void **changed** ()  
Emits when some data changed.

### 10.1.63 Class `sh::actions::common::ActionOpenTerminalAsRoot`

**class** `sh::actions::common::ActionOpenTerminalAsRoot` : **public** `sh::actions::common::ActionOpenTerminalAsU`  
Action for opening a commandline terminal as root.

#### Public Functions

**ActionOpenTerminalAsRoot** (QString *dir*)

void **execute** ()  
Executes this action.

void **execute** (`sh::actions::ActionExecutionInfo` \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).



void **setShortcutHint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

### 10.1.64 Class `sh::actions::common::ActionOpenTerminalAsUser`

**class** `sh::actions::common::ActionOpenTerminalAsUser` : **public** `sh::actions::ActionActionItem`  
Action for opening a commandline terminal (with current user).

Subclassed by `sh::actions::common::ActionOpenTerminalAsRoot`

## Public Functions

**ActionOpenTerminalAsUser** (QString *dir*)

void **execute** ()  
Executes this action.

void **execute** (`sh::actions::ActionExecutionInfo` \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

### 10.1.65 Class `sh::actions::common::ActionRenameItem`

**class** `sh::actions::common::ActionRenameItem` : **public** `sh::actions::ActionActionItem`  
Action for renaming items.

#### Public Functions

**ActionRenameItem** (QList<std::shared\_ptr<[sh::filesystem::FilesystemNode](#)>> *nodes*)

void **execute** ()  
Executes this action.

void **execute** ([sh::actions::ActionExecutionInfo](#) \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

## Public Static Functions

void **doInitialize** ()  
void **doShutdown** ()

### 10.1.66 Class `sh::actions::common::ActionShowProperties`

**class** `sh::actions::common::ActionShowProperties` : **public** `sh::actions::ActionActionItem`  
Action for showing the properties dialog for filesystem nodes.

## Public Functions

**ActionShowProperties** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> *nodes*)

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.  
The action with the highest value becomes the default. Only values >0 will be considered. See *ActionDefaultPrecedenceValues* for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<AbstractActionItem> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<AbstractActionItem> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

## Public Static Functions

void **doInitialize** ()  
void **doShutdown** ()

### 10.1.67 Class sh::actions::common::ActionTransferToCopy

**class** *sh::actions::common::ActionTransferToCopy* : **public** *sh::actions::common::ActionAbstractTransferTo*  
Action for copying a selection to some other locations.

## Public Functions

**ActionTransferToCopy** (QList<std::shared\_ptr<*sh::filesystem::FileSystemNode*>> *nodes*)

**const** QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> **subitems** ()  
Returns the list of subitems from this submenu.

void **setSubitems** (const QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> *subitems*)  
Sets the subitems.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<*AbstractActionItem*> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<*AbstractActionItem*> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void> *> oninitialized* = 0)  
Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **subitemsChanged** ()  
Emits when the list of subitems changed.

void **changed** ()  
Emits when some data changed.

## Public Static Functions

void **doInitialize** ()  
Only called by the Shallot infrastructure for initialization.

void **doShutdown** ()  
Only called by the Shallot infrastructure for initialization.

### 10.1.68 Class `sh::actions::common::ActionTransferToHelper`

**class** `sh::actions::common::ActionTransferToHelper` : **public** `sh::actions::ActionActionItem`  
This action is used as subitems in *ActionAbstractTransferTo*.

## Public Functions

**ActionTransferToHelper** (QString *text*, QString *icon*, std::shared\_ptr<const  
*sh::filesystem::Eurl*> *transferTo*, std::shared\_ptr<ActionAbstractTransferTo>  
*root*)

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.



bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

### 10.1.69 Class `sh::actions::common::ActionTransferToMove`

**class** `sh::actions::common::ActionTransferToMove` : **public** `sh::actions::common::ActionAbstractTransferTo`  
Action for moving a selection to some other locations.

#### Public Functions

**ActionTransferToMove** (`QList<std::shared_ptr<sh::filesystem::FilesystemNode>> nodes`)

**const** `QList<std::shared_ptr<sh::actions::AbstractActionItem>> subitems` ()  
Returns the list of subitems from this submenu.

**void** **setSubitems** (**const** `QList<std::shared_ptr<sh::actions::AbstractActionItem>> subitems`)  
Sets the subitems.

`QString` **text** ()  
Returns the displayed text for this action.

`QString` **icon** ()  
Returns the icon for this action.

**bool** **enabled** ()  
Checks if this action is enabled.

**bool** **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

**bool** **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

**int** **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

**void** **setText** (`QString text`)  
Sets the displayed text.

**void** **setIcon** (`QString icon`)  
Sets the icon.

**void** **setEnabled** (**bool** *enabled*)  
Sets if the item is enabled.

**void** **setChecked** (**bool** *checked*)  
Sets if the item is checked (has a cross in the ui).

**void** **setVisible** (**bool** *visible*)  
Sets the visibility of this item.

**bool** **visible** ()  
Checks the visibility of this item (non-recursively).

`std::weak_ptr<AbstractActionItem>` **parentAction** ()  
Returns the parent action, if it is added to a container.

**void** **\_setParentAction** (`std::shared_ptr<AbstractActionItem>` *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
     > *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
     Synchronously initializes the action.

bool **isInitialized** ()  
     Checks if this action is initialized.

bool **isInitializing** ()  
     Checks if this action is initializing.

## Signals

void **subitemsChanged** ()  
     Emits when the list of subitems changed.

void **changed** ()  
     Emits when some data changed.

## Public Static Functions

void **doInitialize** ()

void **doShutdown** ()

### 10.1.70 Class `sh::actions::mainmenu::ActionAbout`

**class** `sh::actions::mainmenu::ActionAbout` : **public** `sh::actions::ActionActionItem`  
     Action showing the Shallot about box.

## Public Functions

**ActionAbout** ()

void **execute** ()  
     Executes this action.

void **execute** (`sh::actions::ActionExecutionInfo` \**info*)  
     Executes this action.

QKeySequence **shortcuthint** ()  
     Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
     Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
     Sets the keyboard shortcut for triggering this action.

QString **text** ()  
     Returns the displayed text for this action.

QString **icon** ()  
     Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

### 10.1.71 Class `sh::actions::mainmenu::ActionAbstractSelectNodes`

**class** `sh::actions::mainmenu::ActionAbstractSelectNodes` : **public** `sh::actions::ActionActionItem`  
Abstract subclass for actions which select some nodes in the current file view.

Subclassed by `sh::actions::mainmenu::ActionInvertNodeSelection`, `sh::actions::mainmenu::ActionSelectAllNodes`

## Public Functions

**ActionAbstractSelectNodes** (QString *text*)

void **execute** ()  
Executes this action.

void **execute** (`sh::actions::ActionExecutionInfo` \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0)  
Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

### 10.1.72 Class `sh::actions::mainmenu::ActionApplyProfile`

**class** `sh::actions::mainmenu::ActionApplyProfile` : **public** `sh::actions::SubmenuItem`  
Submenu action for switching the current profile.

#### Public Functions

**ActionApplyProfile** ()

**const** `QList<std::shared_ptr<sh::actions::AbstractActionItem>>` **subitems** ()  
Returns the list of subitems from this submenu.

void **setSubitems** (**const** `QList<std::shared_ptr<sh::actions::AbstractActionItem>>` *subitems*)  
Sets the subitems.

`QString` **text** ()  
Returns the displayed text for this action.

`QString` **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **subitemsChanged** ()  
Emits when the list of subitems changed.

void **changed** ()  
Emits when some data changed.

**class ActionApplyProfileX**: public [sh::actions::ActionActionItem](#)  
Action for switching the current profile to a certain one.

## Public Functions

**ActionApplyProfileX** (QString *profilename*, bool *checked*)

void **execute** ()

Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)

Executes this action.

QKeySequence **shortcuthint** ()

Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()

Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)

Sets the keyboard shortcut for triggering this action.

QString **text** ()

Returns the displayed text for this action.

QString **icon** ()

Returns the icon for this action.

bool **enabled** ()

Checks if this action is enabled.

bool **isChecked** ()

Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.



```

void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.

```

### Signals

```

void changed ()
    Emits when some data changed.

```

## 10.1.73 Class `sh::actions::mainmenu::ActionApplyProfile::ActionApplyProfileX`

```

class sh::actions::mainmenu::ActionApplyProfile::ActionApplyProfileX: public sh::actions::ActionAct
    Action for switching the current profile to a certain one.

```

### Public Functions

```

ActionApplyProfileX (QString profilename, bool checked)

void execute ()
    Executes this action.

void execute (sh::actions::ActionExecutionInfo *info)
    Executes this action.

QKeySequence shortcuthint ()
    Returns the keyboard shortcut for triggering this action.

bool shortcuthintTriggersOnCurrentDirectory ()
    Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry
    selection).

void setShortcuthint (QKeySequence shortcut, bool triggersOnCurrentDirectory = false)
    Sets the keyboard shortcut for triggering this action.

QString text ()
    Returns the displayed text for this action.

QString icon ()
    Returns the icon for this action.

bool enabled ()
    Checks if this action is enabled.

bool isChecked ()
    Checks if this action is checked (has a cross in the ui).

bool isCheckable ()
    Checks if this action is checkable (can have a cross in the ui).

```

int **defaultActionPrecedence** () const

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>)

> *oninitialized* = 0 Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **changed** ()

Emits when some data changed.

### 10.1.74 Class `sh::actions::mainmenu::ActionFileViewChangeIconDimension`

**class** `sh::actions::mainmenu::ActionFileViewChangeIconDimension` : **public** `sh::actions::ActionActionItem`  
Action for changing the icon size.

Subclassed by `sh::actions::mainmenu::ActionFileViewDecreaseIconDimension`,  
`sh::actions::mainmenu::ActionFileViewIncreaseIconDimension`

## Public Functions

**ActionFileViewChangeIconDimension** (QString *name*, int *direction*, QString *icon*, QKeySequence *shortcut*)

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<*AbstractActionItem*> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

### 10.1.75 Class `sh::actions::mainmenu::ActionFileViewDecreaseIconDimension`

**class** `sh::actions::mainmenu::ActionFileViewDecreaseIconDimension` : **public** `sh::actions::mainmenu::ActionFileViewDecreaseIconDimension`  
Action for decreasing the icon size.

#### Public Functions

**ActionFileViewDecreaseIconDimension** ()

void **execute** ()  
Executes this action.

void **execute** (`sh::actions::ActionExecutionInfo` \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () const

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>)

> *oninitialized* = 0 Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **changed** ()

Emits when some data changed.

### 10.1.76 Class `sh::actions::mainmenu::ActionFileViewIncreaseIconDimension`

**class** `sh::actions::mainmenu::ActionFileViewIncreaseIconDimension` : **public** `sh::actions::mainmenu::Act`

Action for increasing the icon size.

## Public Functions

**ActionFileViewIncreaseIconDimension** ()

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \*info)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

```
void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.
```

## Signals

```
void changed ()
    Emits when some data changed.
```

### 10.1.77 Class `sh::actions::mainmenu::ActionFileviewFilesizeFormatting`

```
class sh::actions::mainmenu::ActionFileviewFilesizeFormatting: public sh::actions::ActionActionItem
    Action for toggling the filesize formatting mode for the current fileview.
```

## Public Functions

```
ActionFileviewFilesizeFormatting ()

void execute ()
    Executes this action.

void execute (sh::actions::ActionExecutionInfo *info)
    Executes this action.

QKeySequence shortcuthint ()
    Returns the keyboard shortcut for triggering this action.

bool shortcuthintTriggersOnCurrentDirectory ()
    Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry
    selection).

void setShortcuthint (QKeySequence shortcut, bool triggersOnCurrentDirectory = false)
    Sets the keyboard shortcut for triggering this action.

QString text ()
    Returns the displayed text for this action.

QString icon ()
    Returns the icon for this action.

bool enabled ()
    Checks if this action is enabled.

bool isChecked ()
    Checks if this action is checked (has a cross in the ui).

bool isCheckable ()
    Checks if this action is checkable (can have a cross in the ui).
```

int **defaultActionPrecedence** () const

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>)

> *oninitialized* = 0 Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **changed** ()

Emits when some data changed.

### 10.1.78 Class `sh::actions::mainmenu::ActionFileviewShowHiddenFiles`

**class** `sh::actions::mainmenu::ActionFileviewShowHiddenFiles` : **public** `sh::actions::ActionActionItem`

Action which toggles the visibility of hidden files in the current fileview.



## Public Functions

**ActionFileviewShowHiddenFiles** ()

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \*info)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<*AbstractActionItem*> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

### 10.1.79 Class `sh::actions::mainmenu::ActionFileviewView`

**class** `sh::actions::mainmenu::ActionFileviewView` : **public** `sh::actions::ActionActionItem`  
Action which toggles between icon- and listmode for the current fileview.

#### Public Functions

**ActionFileviewView** ()

void **execute** ()  
Executes this action.

void **execute** (`sh::actions::ActionExecutionInfo` \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () const

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>)

> *oninitialized* = 0 Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **changed** ()

Emits when some data changed.

### 10.1.80 Class sh::actions::mainmenu::ActionGotoDirectory

class [sh::actions::mainmenu::ActionGotoDirectory](#) : public [sh::actions::ActionActionItem](#)

Action which toggles the visibility of hidden files in the current fileview.

## Public Functions

### ActionGotoDirectory ()

void **execute** ()

Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo \*info*)

Executes this action.

QKeySequence **shortcuthint** ()

Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()

Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)

Sets the keyboard shortcut for triggering this action.

QString **text** ()

Returns the displayed text for this action.

QString **icon** ()

Returns the icon for this action.

bool **enabled** ()

Checks if this action is enabled.

bool **isChecked** ()

Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

```
void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.
```

## Signals

```
void changed ()
    Emits when some data changed.
```

### 10.1.81 Class `sh::actions::mainmenu::ActionHelp`

```
class sh::actions::mainmenu::ActionHelp : public sh::actions::ActionActionItem
    Action for opening the Shallot help system.
```

#### Public Functions

```
ActionHelp ()

void execute ()
    Executes this action.

void execute (sh::actions::ActionExecutionInfo *info)
    Executes this action.

QKeySequence shortcuthint ()
    Returns the keyboard shortcut for triggering this action.

bool shortcuthintTriggersOnCurrentDirectory ()
    Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry
    selection).

void setShortcuthint (QKeySequence shortcut, bool triggersOnCurrentDirectory = false)
    Sets the keyboard shortcut for triggering this action.

QString text ()
    Returns the displayed text for this action.

QString icon ()
    Returns the icon for this action.

bool enabled ()
    Checks if this action is enabled.

bool isChecked ()
    Checks if this action is checked (has a cross in the ui).

bool isCheckable ()
    Checks if this action is checkable (can have a cross in the ui).
```

int **defaultActionPrecedence** () const

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>)

> *oninitialized* = 0 Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **changed** ()

Emits when some data changed.

### 10.1.82 Class `sh::actions::mainmenu::ActionInvertNodeSelection`

**class** `sh::actions::mainmenu::ActionInvertNodeSelection` : **public** `sh::actions::mainmenu::ActionAbstractSele`

Action for inverting the node selection in the current file view.

## Public Functions

**ActionInvertNodeSelection** ()

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \*info)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<*AbstractActionItem*> parent)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

### Signals

void **changed** ()  
Emits when some data changed.

### Private Members

QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> **wasselected**

## 10.1.83 Class *sh::actions::mainmenu::ActionMain*

**class** *sh::actions::mainmenu::ActionMain* : **public** *sh::actions::SubmenuItem*, **public** *sh::base::Singleton*  
Submenu action providing the Shallot main menu.

### Public Functions

**const** QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> **subitems** ()  
Returns the list of subitems from this submenu.

void **setSubitems** (**const** QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> *subitems*)  
Sets the subitems.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.



The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

```
void setText (QString text)
    Sets the displayed text.

void setIcon (QString icon)
    Sets the icon.

void setEnabled (bool enabled)
    Sets if the item is enabled.

void setChecked (bool checked)
    Sets if the item is checked (has a cross in the ui).

void setVisible (bool visible)
    Sets the visibility of this item.

bool visible ()
    Checks the visibility of this item (non-recursively).

std::weak_ptr<AbstractActionItem> parentAction ()
    Returns the parent action, if it is added to a container.

void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.

void doInitialize ()
    Executes singleton initialization.

void doShutdown ()
    Executes singleton shutdown.

void shutdown ()
    Shutdown down this singleton.

bool isAlive ()
    Returns if this singleton is alive (true until its shutdown begins).
```

## Signals

```
void subitemsChanged ()
    Emits when the list of subitems changed.

void changed ()
    Emits when some data changed.
```

## Private Functions

**ActionMain()**

### 10.1.84 Class `sh::actions::mainmenu::ActionManageSavedSettings`

**class** `sh::actions::mainmenu::ActionManageSavedSettings` : **public** `sh::actions::ActionActionItem`  
Action for opening the 'Manage saved settings' dialog.

## Public Functions

**ActionManageSavedSettings()**

void **execute()**  
Executes this action.

void **execute**(`sh::actions::ActionExecutionInfo *info`)  
Executes this action.

QKeySequence **shortcuthint()**  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory()**  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint**(QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text()**  
Returns the displayed text for this action.

QString **icon()**  
Returns the icon for this action.

bool **enabled()**  
Checks if this action is enabled.

bool **isChecked()**  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable()**  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence()** **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText**(QString *text*)  
Sets the displayed text.

void **setIcon**(QString *icon*)  
Sets the icon.

void **setEnabled**(bool *enabled*)  
Sets if the item is enabled.

```

void setChecked (bool checked)
    Sets if the item is checked (has a cross in the ui).

void setVisible (bool visible)
    Sets the visibility of this item.

bool visible ()
    Checks the visibility of this item (non-recursively).

std::weak_ptr<AbstractActionItem> parentAction ()
    Returns the parent action, if it is added to a container.

void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.

```

## Signals

```

void changed ()
    Emits when some data changed.

```

### 10.1.85 Class `sh::actions::mainmenu::ActionNumberFileviews`

```

class sh::actions::mainmenu::ActionNumberFileviews : public sh::actions::SubmenuItem
    Submenu action for changing the number of fileviews.

```

#### Public Functions

```

ActionNumberFileviews ()

const QList<std::shared_ptr<sh::actions::AbstractActionItem>> subitems ()
    Returns the list of subitems from this submenu.

void setSubitems (const QList<std::shared_ptr<sh::actions::AbstractActionItem>> subitems)
    Sets the subitems.

QString text ()
    Returns the displayed text for this action.

QString icon ()
    Returns the icon for this action.

bool enabled ()
    Checks if this action is enabled.

bool isChecked ()
    Checks if this action is checked (has a cross in the ui).

```

bool **isCheckedable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>

> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **subitemsChanged** ()

Emits when the list of subitems changed.

void **changed** ()

Emits when some data changed.

## Friends

```
friend class ActionNumberFileviews_Add
friend class ActionNumberFileviews_Remove
friend class ActionNumberFileviews_Remove_Current
```

### 10.1.86 Class `sh::actions::mainmenu::ActionNumberFileviews_Add`

**class** `sh::actions::mainmenu::ActionNumberFileviews_Add`: public `sh::actions::ActionActionItem`  
 Action for adding a new fileview.

## Public Functions

```
ActionNumberFileviews_Add (QList<std::shared_ptr<sh::filesystem::FilesystemNode>> nodes =
                             {})
```

void **execute** ()  
 Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \*info)  
 Executes this action.

QKeySequence **shortcuthint** ()  
 Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
 Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence shortcut, bool triggersOnCurrentDirectory = false)  
 Sets the keyboard shortcut for triggering this action.

QString **text** ()  
 Returns the displayed text for this action.

QString **icon** ()  
 Returns the icon for this action.

bool **enabled** ()  
 Checks if this action is enabled.

bool **isChecked** ()  
 Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
 Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
 Returns the precedence for becoming the default action of a parent submenu.  
 This e.g. leads to a bold label.  
 The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString text)  
 Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<AbstractActionItem> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_\_setParentAction** (std::shared\_ptr<AbstractActionItem> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
    > *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

## Public Static Functions

void **doInitialize** ()

void **doShutdown** ()

### 10.1.87 Class sh::actions::mainmenu::ActionNumberFileviews\_Remove

**class** *sh::actions::mainmenu::ActionNumberFileviews\_Remove* : **public** *sh::actions::ActionActionItem*  
Action for removing a certain fileview.

## Public Functions

**ActionNumberFileviews\_Remove** (int *i*, bool *deflt*)

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<*AbstractActionItem*> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

### 10.1.88 Class `sh::actions::mainmenu::ActionNumberFileviews_Remove_Current`

**class** `sh::actions::mainmenu::ActionNumberFileviews_Remove_Current` : public `sh::actions::ActionActionItem`  
Action for removing the current fileview (for keyboard shortcut).

#### Public Functions

**ActionNumberFileviews\_Remove\_Current** (QList<std::shared\_ptr<`sh::filesystem::FilesystemNode`>>  
*nodes*)

void **execute** ()  
Executes this action.

void **execute** (`sh::actions::ActionExecutionInfo` \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).



bool **isCheckedable** ()  
 Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
 Returns the precedence for becoming the default action of a parent submenu.  
 This e.g. leads to a bold label.  
 The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
 Sets the displayed text.

void **setIcon** (QString *icon*)  
 Sets the icon.

void **setEnabled** (bool *enabled*)  
 Sets if the item is enabled.

void **setChecked** (bool *checked*)  
 Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
 Sets the visibility of this item.

bool **visible** ()  
 Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
 Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
 Sets the parent action. .

void **initializeAsync** (std::function<void>  
 > *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
 Synchronously initializes the action.

bool **isInitialized** ()  
 Checks if this action is initialized.

bool **isInitializing** ()  
 Checks if this action is initializing.

## Signals

void **changed** ()  
 Emits when some data changed.

## Public Static Functions

void **doInitialize** ()

void **doShutdown** ()

### 10.1.89 Class `sh::actions::mainmenu::ActionQuit`

**class** `sh::actions::mainmenu::ActionQuit` : **public** `sh::actions::ActionActionItem`  
Action for closing Shallot.

## Public Functions

**ActionQuit** ()

void **execute** ()  
Executes this action.

void **execute** (`sh::actions::ActionExecutionInfo` \*info)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0)  
Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

### 10.1.90 Class `sh::actions::mainmenu::ActionRefresh`

**class** `sh::actions::mainmenu::ActionRefresh` : **public** `sh::actions::ActionActionItem`  
Action for refreshing the current view.

#### Public Functions

**ActionRefresh** ()

void **execute** ()  
Executes this action.

void **execute** (`sh::actions::ActionExecutionInfo` \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

### 10.1.91 Class `sh::actions::mainmenu::ActionSaveSettings`

**class** `sh::actions::mainmenu::ActionSaveSettings` : **public** `sh::actions::ActionActionItem`  
Action for opening the ‘Save settings’ dialog.

#### Public Functions

**ActionSaveSettings** ()

void **execute** ()  
Executes this action.

void **execute** (`sh::actions::ActionExecutionInfo *info`)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0)  
Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

### 10.1.92 Class `sh::actions::mainmenu::ActionSearch`

**class** `sh::actions::mainmenu::ActionSearch` : **public** `sh::actions::ActionActionItem`  
Action for opening the ‘Save settings’ dialog.

#### Public Functions

**ActionSearch** ()

void **execute** ()  
Executes this action.

void **execute** (`sh::actions::ActionExecutionInfo` \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

### 10.1.93 Class `sh::actions::mainmenu::ActionSelectAllNodes`

**class** `sh::actions::mainmenu::ActionSelectAllNodes` : **public** `sh::actions::mainmenu::ActionAbstractSelectNode`  
Action for selecting all nodes in the current file view.

#### Public Functions

**ActionSelectAllNodes** ()

void **execute** ()  
Executes this action.

void **execute** (`sh::actions::ActionExecutionInfo` \*info)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.



void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

### 10.1.94 Class sh::actions::mainmenu::ActionSetWindowTitlePattern

**class** *sh::actions::mainmenu::ActionSetWindowTitlePattern* : **public** *sh::actions::ActionActionItem*  
Action for setting the window title pattern.

#### Public Functions

**ActionSetWindowTitlePattern** ()

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

### 10.1.95 Class `sh::actions::mainmenu::ActionShowDetailPanel`

**class** `sh::actions::mainmenu::ActionShowDetailPanel` : **public** `sh::actions::ActionActionItem`  
Action for toggling the visibility of the detail panel.

#### Public Functions

**ActionShowDetailPanel** ()

void **execute** ()  
Executes this action.

void **execute** (`sh::actions::ActionExecutionInfo` \*info)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

### 10.1.96 Class `sh::actions::mainmenu::ActionShowFolderTree`

**class** `sh::actions::mainmenu::ActionShowFolderTree` : **public** `sh::actions::ActionActionItem`  
Action for toggling the visibility of the directory tree.

#### Public Functions

**ActionShowFolderTree** ()

void **execute** ()  
Executes this action.

void **execute** (`sh::actions::ActionExecutionInfo` \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

### 10.1.97 Class `sh::actions::mainmenu::ActionShowLog`

**class** `sh::actions::mainmenu::ActionShowLog` : **public** `sh::actions::ActionActionItem`  
Action for showing the Shallot log in a dialog.

#### Public Functions

**ActionShowLog** ()

void **execute** ()  
Executes this action.

void **execute** (`sh::actions::ActionExecutionInfo` \*info)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

### 10.1.98 Class `sh::actions::mainmenu::ActionTreeStickyness`

**class** `sh::actions::mainmenu::ActionTreeStickyness` : **public** `sh::actions::ActionActionItem`  
Action for toggling the stickyness of the directory tree to the fileviews.

#### Public Functions

**ActionTreeStickyness** ()

void **execute** ()  
Executes this action.

void **execute** (`sh::actions::ActionExecutionInfo` \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.



## Signals

void **changed** ()  
Emits when some data changed.

### 10.1.99 Class `sh::actions::mainmenu::ActionTuning`

**class** `sh::actions::mainmenu::ActionTuning` : **public** `sh::actions::ActionActionItem`  
Action for opening the ‘Tuning’ dialog.

#### Public Functions

**ActionTuning** ()

void **execute** ()  
Executes this action.

void **execute** (`sh::actions::ActionExecutionInfo` \*info)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

### 10.1.100 Class `sh::base::IconManager`

**class** `sh::base::IconManager` : **public** `QObject`, **public** `sh::base::Singleton`  
Fetches icons according to theme availability and settings.

#### Public Functions

`QIcon` **getIcon** (`QString` *mainname*, `QString` *emblemname* = `QString()`, `QString` *miniemblemname* = `QString()`, `QStringList` *tags* = `QStringList()`)  
Creates a `QIcon` from the icon name(s). Uses a cache.

`QIcon` **getIconByFullname** (`QString` *fullname*)  
Creates a `QIcon` from the icon fullname. Uses a cache.

`QIcon` **getIcon** (`QIcon` *mainicon*, `QString` *emblemname*, `QString` *miniemblemname*, `QStringList` *tags*)  
Creates a `QIcon` based on an existing one. Uncached.

`QPixmap` **getPixmap** (`QString` *name*, int *size* = 22)  
Creates a `QPixmap` for an icon name. Uncached.

**~IconManager** ()

void **doInitialize** ()  
Executes singleton initialization.

```
void doShutdown ()
    Executes singleton shutdown.

void shutdown ()
    Shutdown down this singleton.

bool isAlive ()
    Returns if this singleton is alive (true until its shutdown begins).
```

### Private Functions

```
IconManager ()

QIcon _getIcon (QString name)

QImage _colourImage (QImage img, QColor color)
```

### Private Members

```
QMutex cachemutex

QHash<QString, QIcon> cache

QList<GetIconStrategy*> getIconStrategies

std::shared_ptr<sh::configuration::ConfigurationValue> cfgvalPreferredStrategy
```

### class **GetIconStrategy**

Subclassed by *sh::base::IconManager::IncludedGetIconStrategy*, *sh::base::IconManager::QIconFromThemeGetIconStrategy*

### Public Functions

```
QIcon getIcon (QString name) = 0

~GetIconStrategy ()

class IncludedGetIconStrategy : public sh::base::IconManager::GetIconStrategy
```

### Public Functions

```
IncludedGetIconStrategy ()

QIcon getIcon (QString name)
```

### Private Members

```
QColor brandingcolor1

QColor brandingcolor2

class QIconFromThemeGetIconStrategy : public sh::base::IconManager::GetIconStrategy
```

### Public Functions

`QIconFromThemeGetIconStrategy()`

`QIcon getIcon (QString name)`

### Private Members

`QHash<QString, QString> _aliases`

## 10.1.101 Class `sh::base::IconManager::GetIconStrategy`

**class** `sh::base::IconManager::GetIconStrategy`

Subclassed by `sh::base::IconManager::IncludedGetIconStrategy`, `sh::base::IconManager::QIconFromThemeGetIconStrategy`

### Public Functions

`QIcon getIcon (QString name) = 0`

`~GetIconStrategy()`

## 10.1.102 Class `sh::base::IconManager::IncludedGetIconStrategy`

**class** `sh::base::IconManager::IncludedGetIconStrategy` : **public** `sh::base::IconManager::GetIconStrategy`

### Public Functions

`IncludedGetIconStrategy()`

`QIcon getIcon (QString name)`

### Private Members

`QColor brandingcolor1`

`QColor brandingcolor2`

## 10.1.103 Class `sh::base::IconManager::QIconFromThemeGetIconStrategy`

**class** `sh::base::IconManager::QIconFromThemeGetIconStrategy` : **public** `sh::base::IconManager::GetIconStrategy`

## Public Functions

**QIconFromThemeGetIconStrategy** ()

QIcon **getIcon** (QString *name*)

## Private Members

QHash<QString, QString> **\_aliases**

### 10.1.104 Class sh::base::Logger

**class** *sh::base::Logger* : **public** QObject, **public** *sh::base::Singleton*

The logging manager.

Use it for writing messages to the Shallot log and for reading from it.

Note: Logging is easiest by the LOG\_\* macros like SH\_LOG\_INFO (and it provides more meta data!).

## Public Functions

void **logException** (*sh::exceptions::Exception* &*ex*, *LogSeverity* *severity* = *LogSeverity::ERROR*,  
QString *source* = QString())

Logs an exception.

void **log** (QString *message*, *LogSeverity* *severity*, QString *source* = QString())

Logs a message.

QString **getLogAsText** (*LogSeverity* *minseverity* = *LogSeverity::DEBUG*)

Returns all logged messages as one text.

QList<*LogMessage\**> **logMessages** ()

Returns the list of logged messages.

QString **logPrefix** ()

Returns the log prefix, i.e. an optional string all log output begins with.

**~Logger** ()

void **doInitialize** ()

Executes singleton initialization.

void **doShutdown** ()

Executes singleton shutdown.

void **shutdown** ()

Shutdown down this singleton.

bool **isAlive** ()

Returns if this singleton is alive (true until its shutdown begins).

## Signals

void **newLogMessageArrived** ()  
Emitted when a new log message was written.

## Private Functions

**Logger** ()

## Private Members

QMutex **mutex**  
QList<*LogMessage*\*> **\_logMessages**  
QString **\_logPrefix**  
**struct LogMessage**  
A single log message.

## Public Members

QString **message**  
*LogSeverity* **severity**  
QString **source**  
QDateTime **time**

### 10.1.105 Class `sh::base::MainThread`

**class** *sh::base::MainThread*  
The main thread.

Lots of data structure may only accessed from the main thread. The UI and the *sh::filesystem::FilesystemModel* also live in this thread (although some functions there may allow multi-threading in some ways).

## Public Static Functions

*sh::base::ThreadDispatcher* \***dispatcher** ()  
The *sh::base::ThreadDispatcher*, which allows to execute code in the main thread.

### 10.1.106 Class `sh::base::ShallotProcess`

**class** *sh::base::ShallotProcess* : **public** *sh::base::Singleton*  
Provides parameters given from the user by command line and some more simple infos.

## Public Functions

- QString uiMode ()**  
The ui mode (e.g. “qt”, “web”) specified on command line.
- QMap<QString, QString> configurationValueAssignments ()**  
The configuration value assignments set on command line.
- QList<QString> configurationValueFiles ()**  
The configuration value files set on command line.
- QString parameterValue (QString key, QString deflt = QString())**  
Returns the command line parameter value for a key as string.
- QStringList parameterValueList (QString key)**  
Returns the command line parameter value for a key as list of strings (for multiple parameter usage).
- qint64 parameterValueInt (QString key, qint64 deflt = 0)**  
Returns the command line parameter value for a key as integer.
- QStringList parameters ()**  
Returns the list of keys of all specified command line parameter names.
- QString initialWorkDirectory ()**  
Returns the initial directory specified on command line.
- QString logPrefix ()**  
Returns the log prefix (i.e. an optional string all log output begins with) specified on command line.
- QStringList lang ()**  
Returns the ui language specified on command line.
- QColor brandingColor ()**  
Returns the Shallot branding color (typically a dark red).
- int minport ()**  
Return the minimal allowed port specified on command line (for web ui).
- int maxport ()**  
Return the maximal allowed port specified on command line (for web ui).
- int workerminport ()**  
Return the minimal allowed port specified on command line for spawning web workers with (for web ui).
- int workermaxport ()**  
Return the maximal allowed port specified on command line for spawning web workers with (for web ui).
- int maxnumberworkers (int deflt)**  
Maximum number of workers specified on command line.
- int maxnumberworkersperhost (int deflt)**  
Maximum number of workers per host specified on command line.
- int maxmemorypercent (int deflt)**  
Maximum memory in percent specified on command line.
- int maxmemorypercentglobal (int deflt)**  
Maximum memory in percent globally specified on command line.
- int maxbrowserawayseconds (int deflt)**  
Maximum browser away time in seconds specified on command line.
- int maxidlenessesseconds (int deflt)**  
Maximum user idleness time in seconds specified on command line.

void **doInitialize** ()  
Executes singleton initialization.

void **doShutdown** ()  
Executes singleton shutdown.

void **shutdown** ()  
Shutdown down this singleton.

bool **isAlive** ()  
Returns if this singleton is alive (true until its shutdown begins).

## Public Static Functions

QString **userDataDir** ()  
Returns the path to a storage for per-user data. It resides somewhere within the user home directory.

QString **shallotDataDir** ()  
Returns the directory of the Shallot program data. It contains the static files which are part of the Shallot program.

QString **revisionString** ()  
Returns the Shallot revision string, i.e. the exact version number. Note: This is only updated by a special build tool (anise) and not by directly compiling via qmake.

QString **homepage** ()  
Returns the url to the Shallot homepage.

QDateTime **buildtime** ()  
Returns the time when this Shallot build was created. Note: This is only updated by a special build tool (anise) and not by directly compiling via qmake.

## Private Functions

**ShallotProcess** ()

## Private Members

QMutex **\_brandingcolormutex**

QMap<QString, QStringList> **\_cmdlineargs**

QString **\_workdir**

QString **\_ui**

QMap<QString, QString> **\_cfgvalassignments**

QList<QString> **\_cfgvalfiles**

QColor **\_brandingcolor**

std::shared\_ptr<*sh::configuration::ConfigurationValue*> **cfgvalBrandingColor**

int **\_minport** = 0

int **\_maxport** = 0

int **\_workerminport** = 0

int **\_workermaxport** = 0



### 10.1.107 Class `sh::base::Singleton`

**class** `sh::base::Singleton`

A singleton with initialization on Shallot startup and shutdown in the end.

See *SingletonInitializer*.

Subclassed by `sh::actions::ActionsManager`, `sh::actions::mainmenu::ActionMain`, `sh::base::IconManager`, `sh::base::Logger`, `sh::base::ShallotProcess`, `sh::configuration::ConfigurationManager`, `sh::detailcolumns::DetailColumnCustomAttributes`, `sh::detailcolumns::DetailColumnDirectSymlinkTarget`, `sh::detailcolumns::DetailColumnExtendedAttributes`, `sh::detailcolumns::DetailColumnFileSize`, `sh::detailcolumns::DetailColumnMimetype`, `sh::detailcolumns::DetailColumnMtime`, `sh::detailcolumns::DetailColumnResolvedSymlink`, `sh::exceptions::ExceptionHandlerSettingsManager`, `sh::filesystem::FilesystemHandlerRegister`, `sh::filesystem::FilesystemModel`, `sh::filesystem::FilesystemModelDirectoryTreeProxy`, `sh::filesystem::FilesystemModelDirectoryTreeProxyVisibilityEnforcements`, `sh::filesystemhandlers::GnomeIODevicesFilesystemHandler`, `sh::filesystemhandlers::GnomeIONetworkFilesystemHandler`, `sh::filesystemhandlers::GnomeIOSmbFilesystemHandler`, `sh::filesystemhandlers::LocalFilesystemHandler`, `sh::filesystemhandlers::SharcFilesystemHandler`, `sh::paneldetails::PanelDetailManager`, `sh::scripting::api::ApiGlobalObject`, `sh::scripting::PythonScriptInterpreter`, `sh::scripting::ScriptingEngine`, `sh::search::SearchFilesystemHandler`, `sh::search::SearchManager`, `sh::settings::SettingsManager`, `sh::tools::accounts::AccountsManager`, `sh::tools::Benchmarking`, `sh::tools::BookmarkManager`, `sh::tools::DataExchange`, `sh::tools::filetypes::FileTypeManager`, `sh::tools::filetypes::UserDefinedOpenMethodDeterminationStrategy`, `sh::tools::LocalFilesystemWatcher`, `sh::tools::LocalFilesystemWatcherConnector`, `sh::tools::OperationsCache`, `sh::tools::ThumbnailManager`, `sh::tools::VisibleViews`

#### Public Functions

void **doInitialize** ()

Executes singleton initialization.

void **doShutdown** ()

Executes singleton shutdown.

void **shutdown** ()

Shutdown down this singleton.

bool **isAlive** ()

Returns if this singleton is alive (true until its shutdown begins).

**~Singleton** ()

**Singleton** (const *Singleton*&) = delete

**Singleton** (*Singleton*&&) = delete

#### Private Members

QMutex **\_mutex\_shutdown**

bool **\_shutdown** = false

### 10.1.108 Class `sh::base::SingletonInitializer`

**class** `sh::base::SingletonInitializer`

Takes care of initialization and shutdown of infrastructure singletons.

It is a very early part of shallot core infrastructure. Singletons typically participate by using those macros:

`DECLARE_SINGLETON(STYPE)`: Used inside the singleton class definition. `STYPE` is the singleton's class. It must be a subclass of `sh::base::Singleton`. It will get a static `std::shared_ptr<STYPE> instance()` method by that.

`REGISTER_SINGLETON(NS, STYPE, GROUPNAME)`: Used inside the source file. `NS` is the namespace of the singleton. `STYPE` is the singleton's class. `GROUPNAME` is the singleton group name (used for dependency handling, see later).

For singleton groups, there is `REGISTER_SINGLETON_GROUP(GROUPNAME, ...)`: `GROUPNAME` is the group name used for grouping singletons together. Additional arguments are other group names; those groups are considered as dependencies, which must be fulfilled before this group can.

For just initializing stuff via static methods (without a singleton instance), use `REGISTER_STATICINIT(NS, STYPE, GROUPNAME)`: `NS` is the namespace of the class to initialize. `STYPE` is the class to initialize. It must provide public static `void doInitialize()` and static `void doShutdown()`. `GROUPNAME` is the group name.

#### Public Functions

`void callInitializers()`

Executes all create-callbacks at first, then all init-callbacks. In both runs, the callbacks with lower index come first. Afterwards, the singletons are considered as up and running.

`SingletonInitializer()`

Constructed only by the infrastructure and made available otherwise.

`~SingletonInitializer()`

`void shutdown()`

Executes all shutdown-callbacks at first, then all remove-callbacks. The callback order is the reversed one of `callInitializers`. Afterwards, the singletons are considered as shut down and removed.

#### Public Static Functions

`char registerGroup(QString groupname, QStringList groupdeps)`

Registers a singleton group.

You should typically use `REGISTER_SINGLETON_GROUP`. See [SingletonInitializer](#).

`char registerSingleton(QString name, std::function<std::shared_ptr<Singleton>>()`

`> instance) QString groupname` Registers a singleton.

You should typically use `DECLARE_SINGLETON` and `REGISTER_SINGLETON`. See [SingletonInitializer](#).

`QThread *initializerThread()`

`bool isShutdown()`

If the singletons are completely shut down and removed.

### Private Types

```
std::tuple< QString, QString, std::function< std::shared_ptr< Singleton >>> > Singleton
typedef std::tuple<QString, QStringList> SingletonGroupTuple
```

### Private Static Functions

```
bool dependsOn (QString group, QString depgroup)
```

### Private Static Attributes

```
QList<SingletonTuple> *_singletons
QHash<QString, SingletonGroupTuple> *_groups
bool _isshutdown = false
QMutex shutdownmutex
```

## 10.1.109 Class sh::base::ThreadDispatcher

```
class sh::base::ThreadDispatcher : public QObject
    Dispatcher for sync/async invocation of functions into the associated thread.
```

### Public Functions

```
ThreadDispatcher (QObject *parent, QThread *thread)
    Constructed only by the infrastructure and made available otherwise.
```

```
void invokeSync (std::function<void>)
    >Synchronously executes a function in the dispatcher's thread.
```

If it already runs in the current thread, it just executes the function. It returns when the function is completely executed.

```
void invokeAsync (std::function<void>)
    >Asynchronously executes a function in the dispatcher's thread.
```

It directly returns. The actual execution will take place in a later message loop iteration of that thread.

```
void invokeAsync (std::function<void>)
    >std::shared_ptr<void> obj1, std::shared_ptr<void> obj2 = 0, std::shared_ptr<void> obj3 = 0Like the other
    variant, but it can also conserve some shared pointers until after execution.
```

```
bool inThread ()
    Returns if the current thread is already equal to the dispatcher's thread (so dispatching isn't required).
```

## Signals

void **\_invoked** ()

## Private Functions

void **\_invokeAsync** (std::function<void>  
>

## Private Members

QMutex **callsmutex**

QThread \***\_thread**

QQueue<std::function<void ()>> **\_queue**

## Private Slots

void **slot\_invoked** ()

### 10.1.110 Class `sh::base::ThreadPool`

**class** `sh::base::ThreadPool` : **public** QObject

A pool of worker threads doing some short jobs from a queue.

#### Public Static Functions

void **enqueueForce** (std::function<void>  
> *fcn*) Enqueues code to the threadpool.

void **enqueueForce** (std::function<void>  
> *fcn*, std::shared\_ptr<void> *obj1*, std::shared\_ptr<void> *obj2* = 0, std::shared\_ptr<void> *obj3* = 0) Enqueues code to the threadpool.

It can also conserve some shared pointers until after execution.

void **enqueueForceBeyondAsyncCallBarrier** (std::function<void>  
> *fcn*, std::shared\_ptr<void> *obj1* = 0, std::shared\_ptr<void> *obj2* = 0, std::shared\_ptr<void> *obj3* = 0) Enqueues code to the threadpool and forbids async calls within it. This is useful for ensuring that no code might access a certain object after its deletion.

It can also conserve some shared pointers until after execution.

void **enqueueIfIn** (std::function<void>  
> *fcn*, QThread \**thread*) Enqueues code to the threadpool if currently in *thread*. Otherwise executes directly.

void **enqueueIfInMain** (std::function<void>  
> *fcn*) Enqueues code to the threadpool if currently in main thread. Otherwise executes directly.

void **doShutdown** ()  
Only called by the Shallot infrastructure for shutdown.

bool **isShuttingDown** ()

```

bool isShutdown ()
int getThreadCount ()
void respectThreadAbort ()
void doInitialize ()

```

### Private Static Functions

```

void _enqueueForce (std::function<void()>
    > fcn

```

### Private Static Attributes

```

const int THREAD_COUNT = 10
QMutex poolmutex
QWaitCondition pooladdedcondition
QList<std::function<void ()>> tasks
bool _shutdown = false
QList<ThreadPoolThread*> _threads
int _threadsalive = 0

```

### Friends

```

friend class ThreadPoolThread

```

## 10.1.111 Class `sh::base::ThreadPoolThread`

```

class sh::base::ThreadPoolThread : public QThread
    A thread in the thread pool.

```

### Friends

```

friend class ThreadPool

```

## 10.1.112 Class `sh::configuration::ConfigurationManager`

```

class sh::configuration::ConfigurationManager : public QObject, public sh::base::Singleton
    Manages the Shallot configuration.

```

This is the more static part of Shallot. Much more stuff, i.e. everything you see in ‘Manage saved settings’, is in *sh::settings::SettingsManager*.

## Public Functions

```
std::shared_ptr<ConfigurationValue> registerConfigValue (QString name, QVariant deflt, ConfigurationValueType *valuetype = 0,
                                                         QString desc = "", ConfigurationCategory cat = CategoryNone, QString
                                                         longdesc = "", QString changehint =
                                                         "")
```

Creates and registers a new *ConfigurationValue*. This is typically done once at the beginning. Each registered instance is shown in the ‘Tuning’ dialog and can be set (for making changes) and observed (for applying changes) in code as well. Useful e.g. for some internal bookkeeping or for exotic machine-wide configuration values (path to some tool, ...).

```
QList<std::shared_ptr<ConfigurationValue>> getAllConfigurationValues ()
```

Returns a list of all registered configuration value instances.

```
void doInitialize ()
```

Executes singleton initialization.

```
void doShutdown ()
```

Executes singleton shutdown.

```
void shutdown ()
```

Shutdown down this singleton.

```
bool isAlive ()
```

Returns if this singleton is alive (true until its shutdown begins).

## Private Functions

```
QVariant getFixedConfigValue (QString key)
```

Returns a non-null value if ‘key’ was set in a fixed way (e.g. via command line).

```
ConfigurationManager ()
```

## Private Members

```
QHash<QString, std::shared_ptr<ConfigurationValue>> _configvalues
```

```
QHash<QString, QVariant> _fixed_configvalues
```

```
bool _fixed_configvalues_initialized = false
```

```
QMutex _mutex
```

```
class ConfigurationValueImpl : public sh::configuration::ConfigurationValue
```

## Public Functions

```
ConfigurationValueImpl (QString name, QVariant deflt, QString desc, QString longdesc,
                        ConfigurationCategory cat, ConfigurationValueType *valuetype,
                        QString changehint, ConfigurationManager *mgr)
```

```
void setConfigValue (QVariant value) override
```

Sets a new configuration value.

```
QVariant getConfigValueVariant () override
```

Returns the current value of this instance as variant.

**bool mayChange () override**  
Returns if changes are allowed.

**int getConfigValueInt ()**  
Returns the current value of this instance as integer.

**double getConfigValueFloat ()**  
Returns the current value of this instance as floating point number.

**bool getConfigValueBool ()**  
Returns the current value of this instance as boolean.

**QString getConfigValueString ()**  
Returns the current value of this instance as string.

**void consumeValue (QObject \*o, std::function<void> > h)**  
Adds a consumer function, triggered directly and whenever the value changes.

**Parameters**

- *o*: For lifetime monitoring.

**QString name ()**  
Returns the internal name of this configuration.

**QString description ()**  
Returns the short description of this configuration.

**QString longDescription ()**  
Returns the long description of this configuration.

**ConfigurationCategory category ()**  
Returns the category of this configuration (mainly for UI structuring).

**QString changeHint ()**  
Returns the change hint text of this configuration.

**QVariant defaultValue ()**  
Returns the default value of this configuration.

**ConfigurationValueType \*valueType ()**  
Returns the value type of this configuration.

## Public Static Functions

**ConfigurationValueType \*valueTypeString ()**  
Returns the string type for configuration values.

**ConfigurationValueType \*valueTypeInteger ()**  
Returns the integer type for configuration values.

**ConfigurationValueType \*valueTypeFloat ()**  
Returns the floating point number type for configuration values.

**ConfigurationValueType \*valueTypeBoolean ()**  
Returns the boolean type for configuration values.

**ConfigurationValueType \*valueTypeLocalFilePath ()**  
Returns the filepath type for configuration values.

## Private Members

*ConfigurationManager* \*mgr

### 10.1.113 Class sh::configuration::ConfigurationManager::ConfigurationValueImpl

```
class sh::configuration::ConfigurationManager::ConfigurationValueImpl : public sh::configuration::C
```

#### Public Functions

**ConfigurationValueImpl** (QString name, QVariant deflt, QString desc, QString longdesc, *ConfigurationCategory* cat, *ConfigurationValueType* \*valuetype, QString changehint, *ConfigurationManager* \*mgr)

void **setConfigValue** (QVariant value) **override**  
Sets a new configuration value.

QVariant **getConfigValueVariant** () **override**  
Returns the current value of this instance as variant.

bool **mayChange** () **override**  
Returns if changes are allowed.

int **getConfigValueInt** ()  
Returns the current value of this instance as integer.

double **getConfigValueFloat** ()  
Returns the current value of this instance as floating point number.

bool **getConfigValueBool** ()  
Returns the current value of this instance as boolean.

QString **getConfigValueString** ()  
Returns the current value of this instance as string.

void **consumeValue** (QObject \*o, std::function<void>  
> h) Adds a consumer function, triggered directly and whenever the value changes.

#### Parameters

- o: For lifetime monitoring.

QString **name** ()  
Returns the internal name of this configuration.

QString **description** ()  
Returns the short description of this configuration.

QString **longDescription** ()  
Returns the long description of this configuration.

*ConfigurationCategory* **category** ()  
Returns the category of this configuration (mainly for UI structuring).

QString **changeHint** ()  
Returns the change hint text of this configuration.

QVariant **defaultValue** ()  
Returns the default value of this configuration.



*ConfigurationValueType* \***valueType** ()  
Returns the value type of this configuration.

### Public Static Functions

*ConfigurationValueType* \***valueTypeString** ()  
Returns the string type for configuration values.

*ConfigurationValueType* \***valueTypeInteger** ()  
Returns the integer type for configuration values.

*ConfigurationValueType* \***valueTypeFloat** ()  
Returns the floating point number type for configuration values.

*ConfigurationValueType* \***valueTypeBoolean** ()  
Returns the boolean type for configuration values.

*ConfigurationValueType* \***valueTypeLocalFilePath** ()  
Returns the filepath type for configuration values.

### Private Members

*ConfigurationManager* \*mgr

## 10.1.114 Class sh::configuration::ConfigurationValue

**class** *sh::configuration::ConfigurationValue*

Abstract base class for a configuration value which is managed in the *Tuning* dialog.

Each instance can get and set the value associated to it.

You should typically not need to override it. See *sh::configuration::ConfigurationManager*. See Shallot documentation for more details.

Subclassed by *sh::configuration::ConfigurationManager::ConfigurationValueImpl*

### Public Functions

**ConfigurationValue** (QString *name*, QVariant *deft*, QString *desc*, QString *longdesc*, *ConfigurationCategory* *cat*, *ConfigurationValueType* \**valuetype*, QString *changehint*)

int **getConfigValueInt** ()  
Returns the current value of this instance as integer.

double **getConfigValueFloat** ()  
Returns the current value of this instance as floating point number.

bool **getConfigValueBool** ()  
Returns the current value of this instance as boolean.

QString **getConfigValueString** ()  
Returns the current value of this instance as string.

QVariant **getConfigValueVariant** () = 0  
Returns the current value of this instance as variant.

void **setConfigValue** (QVariant *value*) = 0  
Sets a new configuration value.

bool **mayChange** () = 0

Returns if changes are allowed.

void **consumeValue** (QObject \*o, std::function<void>

> hAdds a consumer function, triggered directly and whenever the value changes.

#### Parameters

- o: For lifetime monitoring.

QString **name** ()

Returns the internal name of this configuration.

QString **description** ()

Returns the short description of this configuration.

QString **longDescription** ()

Returns the long description of this configuration.

*ConfigurationCategory* **category** ()

Returns the category of this configuration (mainly for UI structuring).

QString **changeHint** ()

Returns the change hint text of this configuration.

QVariant **defaultValue** ()

Returns the default value of this configuration.

*ConfigurationValueType* \***valueType** ()

Returns the value type of this configuration.

#### Public Static Functions

*ConfigurationValueType* \***valueTypeString** ()

Returns the string type for configuration values.

*ConfigurationValueType* \***valueTypeInteger** ()

Returns the integer type for configuration values.

*ConfigurationValueType* \***valueTypeFloat** ()

Returns the floating point number type for configuration values.

*ConfigurationValueType* \***valueTypeBoolean** ()

Returns the boolean type for configuration values.

*ConfigurationValueType* \***valueTypeLocalFilePath** ()

Returns the filepath type for configuration values.

#### Friends

**friend class** ConfigurationManager

**friend class** ConfigurationValueType

### 10.1.115 Class `sh::configuration::ConfigurationValueType`

**class** `sh::configuration::ConfigurationValueType`

Abstract base class for a configuration value type.

Each subclass implements support for a certain type of configuration values (e.g. string, integer, ...).

Subclassed by `sh::configuration::ConfigurationValueTypeBoolean`, `sh::configuration::ConfigurationValueTypeFloat`, `sh::configuration::ConfigurationValueTypeInteger`, `sh::configuration::ConfigurationValueTypeLocalFilePath`, `sh::configuration::ConfigurationValueTypeString`

#### Public Functions

QString **valueDescription** (QVariant *v*)

QVariant **parse** (QString *input*)

### 10.1.116 Class `sh::configuration::ConfigurationValueTypeBoolean`

**class** `sh::configuration::ConfigurationValueTypeBoolean` : **public** `sh::configuration::ConfigurationValueType`

Configuration value type 'boolean'.

#### Public Functions

QString **valueDescription** (QVariant *v*)

QVariant **parse** (QString *input*)

### 10.1.117 Class `sh::configuration::ConfigurationValueTypeFloat`

**class** `sh::configuration::ConfigurationValueTypeFloat` : **public** `sh::configuration::ConfigurationValueType`

Configuration value type 'float'.

#### Public Functions

QString **valueDescription** (QVariant *v*)

QVariant **parse** (QString *input*)

### 10.1.118 Class `sh::configuration::ConfigurationValueTypeInteger`

**class** `sh::configuration::ConfigurationValueTypeInteger` : **public** `sh::configuration::ConfigurationValueType`

Configuration value type 'integer'.

### Public Functions

QString **valueDescription** (QVariant *v*)

QVariant **parse** (QString *input*)

## 10.1.119 Class `sh::configuration::ConfigurationValueTypeLocalFilePath`

**class** `sh::configuration::ConfigurationValueTypeLocalFilePath` : **public** `sh::configuration::ConfigurationValueType`  
Configuration value type ‘file path’.

### Public Functions

QString **valueDescription** (QVariant *v*)

QVariant **parse** (QString *input*)

## 10.1.120 Class `sh::configuration::ConfigurationValueTypeString`

**class** `sh::configuration::ConfigurationValueTypeString` : **public** `sh::configuration::ConfigurationValueType`  
Configuration value type ‘string’.

### Public Functions

QString **valueDescription** (QVariant *v*)

QVariant **parse** (QString *input*)

## 10.1.121 Class `sh::detailcolumns::DetailColumnCustomAttributes`

**class** `sh::detailcolumns::DetailColumnCustomAttributes` : **public** `sh::filesystem::DetailColumn`, **public** `sh::detailcolumns::DetailColumn`  
Detail column which determines the custom attributes.  
Custom attributes are filesystem handler specific values about files (like permissions).

### Public Functions

void **applyValueByEurl** (std::shared\_ptr<const `sh::filesystem::Eurl`> *eurl*,  
`sh::filesystem::Operation` \**op*, QVariant *value*) **override**

QString **name** ()  
The internal unique name.

QString **displayName** ()  
The display name.

bool **isValueAvailable** (std::shared\_ptr<FilesystemNode> *node*)  
Checks if a value for this detail is available for one given node.

QVariant **value** (std::shared\_ptr<FilesystemNode> *node*, bool *ignoreAged* = false)  
Returns the value for this detail for one given node.

### Parameters

- `ignoreAged`: If no value should be returned which is older than the last request (not useful for nearly all cases).

**QString displayValue** (std::shared\_ptr<FilesystemNode> *node*, **const** *sh::filesystem::FilesystemModelFileviewProxy \*viewmodel*)  
Returns the stringified value for this details for one given node considering the configuration of a view.

**QVariant requestValue** (std::shared\_ptr<FilesystemNode> *node*, *sh::filesystem::Operation \*op* = 0, *bool withNodeValues* = false, *bool withOperationsCache* = true)  
Requests to determine the value for this detail for one given node. Blocks until the value is available. Get it with the value method afterwards.

**bool sort\_doTypediff** ()  
If sorting should separate files and directories.

**int sort\_order** (std::shared\_ptr<FilesystemNode> *n1*, std::shared\_ptr<FilesystemNode> *n2*)  
The sorting order.

**uint displayIndex** ()  
Controls which place this column should get in the user interface.

**QVariant computeValue** (std::shared\_ptr<FilesystemNode> *node*, *sh::filesystem::Operation \*op*)  
Returns a freshly determined value for this column and the given node. It neither asks nor populates any caches or storages.

**int defaultWidth** ()

**bool isRightAligned** ()

**void applyValueByEurl** (std::shared\_ptr<**const** *sh::filesystem::Eurl*> *eurl*, *sh::filesystem::Operation \*op*, *QVariant value*)  
Applies the detail value to a given eurl (without using any caches).  
Used for transferring some details when a file transfer occurs.  
Optionally implement the one of the `applyValueBy...` methods and register the instance with *sh::filesystem::FilesystemOperation::addTransferrableDetailColumn*. For performance reasons, you should decide to implement `applyValueByEurl`.

**void applyValueByNode** (std::shared\_ptr<FilesystemNode> *node*, *sh::filesystem::Operation \*op*, *QVariant value*)  
Applies the detail value to a given node (without using any caches).  
Used for transferring some details when a file transfer occurs.  
Optionally implement the one of the `applyValueBy...` methods and register the instance with *sh::filesystem::FilesystemOperation::addTransferrableDetailColumn*. For performance reasons, you should decide to implement `applyValueByEurl`.

**void doShutdown** ()  
Executes singleton shutdown.

**void shutdown** ()  
Shutdown down this singleton.

**bool isAlive** ()  
Returns if this singleton is alive (true until its shutdown begins).

### Public Static Functions

QMap<QString, QString> **getMapByValue** (QVariant *val*)  
Converts the QVariant(QString) from native detail column representation to a QString/QString map.

QVariant **VALUE\_UNAVAILABLE** ()  
A special value expressing that the value is not yet available.

void **registerKnownDetailColumn** (QString *name*, std::shared\_ptr<DetailColumn> *column*)

std::shared\_ptr<DetailColumn> **findKnownDetailColumn** (QString *name*)

### Public Static Attributes

const uint **DISPLAYINDEX\_CORE** = 10000

const uint **DISPLAYINDEX\_INTERESTING** = 20000

const uint **DISPLAYINDEX\_EXOTIC** = 30000

### Private Functions

**DetailColumnCustomAttributes** ()

## 10.1.122 Class `sh::detailcolumns::DetailColumnDirectSymlinkTarget`

**class** `sh::detailcolumns::DetailColumnDirectSymlinkTarget` : **public** `sh::filesystem::DetailColumn`, **public** `DetailColumn` which determines the direct symlink target (by resolving a link once).

### Public Functions

QString **name** ()  
The internal unique name.

QString **displayName** ()  
The display name.

bool **isValueAvailable** (std::shared\_ptr<FilesystemNode> *node*)  
Checks if a value for this detail is available for one given node.

QVariant **value** (std::shared\_ptr<FilesystemNode> *node*, bool *ignoreAged* = false)  
Returns the value for this detail for one given node.

### Parameters

- *ignoreAged*: If no value should be returned which is older than the last request (not useful for nearly all cases).

QString **displayValue** (std::shared\_ptr<FilesystemNode> *node*, `sh::filesystem::FilesystemModelFileviewProxy` \**viewmodel*) **const**  
Returns the stringified value for this details for one given node considering the configuration of a view.

QVariant **requestValue** (std::shared\_ptr<FilesystemNode> *node*, `sh::filesystem::Operation` \**op* = 0, bool *withNodeValues* = false, bool *withOperationsCache* = true)  
Requests to determine the value for this detail for one given node. Blocks until the value is available. Get it with the value method afterwards.

bool **sort\_doTypediff** ()

If sorting should separate files and directories.

int **sort\_order** (std::shared\_ptr<FilesystemNode> *n1*, std::shared\_ptr<FilesystemNode> *n2*)

The sorting order.

uint **displayIndex** ()

Controls which place this column should get in the user interface.

QVariant **computeValue** (std::shared\_ptr<FilesystemNode> *node*, *sh::filesystem::Operation* \**op*)

Returns a freshly determined value for this column and the given node. It neither asks nor populates any caches or storages.

int **defaultWidth** ()

bool **isRightAligned** ()

void **applyValueByEurl** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*,  
*sh::filesystem::Operation* \**op*, QVariant *value*)

Applies the detail value to a given eurl (without using any caches).

Used for transferring some details when a file transfer occurs.

Optionally implement the one of the `applyValueBy...` methods and register the instance with *sh::filesystem::FilesystemOperation::addTransferrableDetailColumn*. For performance reasons, you should decide to implement `applyValueByEurl`.

void **applyValueByNode** (std::shared\_ptr<FilesystemNode> *node*, *sh::filesystem::Operation* \**op*,  
QVariant *value*)

Applies the detail value to a given node (without using any caches).

Used for transferring some details when a file transfer occurs.

Optionally implement the one of the `applyValueBy...` methods and register the instance with *sh::filesystem::FilesystemOperation::addTransferrableDetailColumn*. For performance reasons, you should decide to implement `applyValueByEurl`.

void **doShutdown** ()

Executes singleton shutdown.

void **shutdown** ()

Shutdown down this singleton.

bool **isAlive** ()

Returns if this singleton is alive (true until its shutdown begins).

## Public Static Functions

QVariant **VALUE\_UNAVAILABLE** ()

A special value expressing that the value is not yet available.

void **registerKnownDetailColumn** (QString *name*, std::shared\_ptr<DetailColumn> *column*)

std::shared\_ptr<DetailColumn> **findKnownDetailColumn** (QString *name*)

### Public Static Attributes

```
const uint DISPLAYINDEX_CORE = 10000
const uint DISPLAYINDEX_INTERESTING = 20000
const uint DISPLAYINDEX_EXOTIC = 30000
```

### Private Functions

```
DetailColumnDirectSymlinkTarget ()
```

## 10.1.123 Class `sh::detailcolumns::DetailColumnExtendedAttributes`

```
class sh::detailcolumns::DetailColumnExtendedAttributes : public sh::filesystem::DetailColumn, public sh::detailcolumns::DetailColumn
    Detail column which determines the extended attributes.
```

Extended attributes are a feature of some filesystems.

### Public Functions

```
void applyValueByEurl (std::shared_ptr<const sh::filesystem::Eurl> eurl,
                      sh::filesystem::Operation *op, QVariant value) override
```

```
QString name ()
    The internal unique name.
```

```
QString displayName ()
    The display name.
```

```
bool isValueAvailable (std::shared_ptr<FilesystemNode> node)
    Checks if a value for this detail is available for one given node.
```

```
QVariant value (std::shared_ptr<FilesystemNode> node, bool ignoreAged = false)
    Returns the value for this detail for one given node.
```

#### Parameters

- `ignoreAged`: If no value should be returned which is older than the last request (not useful for nearly all cases).

```
QString displayValue (std::shared_ptr<FilesystemNode> node, const
                     sh::filesystem::FilesystemModelFileviewProxy *viewmodel)
    Returns the stringified value for this details for one given node considering the configuration of a view.
```

```
QVariant requestValue (std::shared_ptr<FilesystemNode> node, sh::filesystem::Operation *op = 0,
                      bool withNodeValues = false, bool withOperationsCache = true)
    Requests to determine the value for this detail for one given node. Blocks until the value is available. Get it with the value method afterwards.
```

```
bool sort_doTypediff ()
    If sorting should separate files and directories.
```

```
int sort_order (std::shared_ptr<FilesystemNode> n1, std::shared_ptr<FilesystemNode> n2)
    The sorting order.
```

```
uint displayIndex ()
    Controls which place this column should get in the user interface.
```



**QVariant computeValue** (std::shared\_ptr<FilesystemNode> *node*, *sh::filesystem::Operation* \**op*)  
Returns a freshly determined value for this column and the given node. It neither asks nor populates any caches or storages.

int **defaultWidth** ()

bool **isRightAligned** ()

void **applyValueByEurl** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*,  
*sh::filesystem::Operation* \**op*, QVariant *value*)  
Applies the detail value to a given eurl (without using any caches).

Used for transferring some details when a file transfer occurs.

Optionally implement the one of the applyValueBy... methods and register the instance with *sh::filesystem::FilesystemOperation::addTransferrableDetailColumn*. For performance reasons, you should decide to implement applyValueByEurl.

void **applyValueByNode** (std::shared\_ptr<FilesystemNode> *node*, *sh::filesystem::Operation* \**op*,  
QVariant *value*)  
Applies the detail value to a given node (without using any caches).

Used for transferring some details when a file transfer occurs.

Optionally implement the one of the applyValueBy... methods and register the instance with *sh::filesystem::FilesystemOperation::addTransferrableDetailColumn*. For performance reasons, you should decide to implement applyValueByEurl.

void **doShutdown** ()  
Executes singleton shutdown.

void **shutdown** ()  
Shutdown down this singleton.

bool **isAlive** ()  
Returns if this singleton is alive (true until its shutdown begins).

## Public Static Functions

QMap<QString, QByteArray> **getMapByValue** (QVariant *val*)  
Converts the QVariant(QString) from native detail column representation to a QString/QString map.

QVariant **VALUE\_UNAVAILABLE** ()  
A special value expressing that the value is not yet available.

void **registerKnownDetailColumn** (QString *name*, std::shared\_ptr<DetailColumn> *column*)  
std::shared\_ptr<DetailColumn> **findKnownDetailColumn** (QString *name*)

## Public Static Attributes

const uint **DISPLAYINDEX\_CORE** = 10000  
const uint **DISPLAYINDEX\_INTERESTING** = 20000  
const uint **DISPLAYINDEX\_EXOTIC** = 30000

## Private Functions

**DetailColumnExtendedAttributes()**

### 10.1.124 Class `sh::detailcolumns::DetailColumnFilesize`

**class** `sh::detailcolumns::DetailColumnFilesize` : **public** `sh::filesystem::DetailColumn`, **public** `sh::base::Singleton`  
Detail column which determines the file size.

## Public Functions

**QString name()**  
The internal unique name.

**QString displayName()**  
The display name.

**bool isValueAvailable**(`std::shared_ptr<FilesystemNode> node`)  
Checks if a value for this detail is available for one given node.

**QVariant value**(`std::shared_ptr<FilesystemNode> node`, `bool ignoreAged = false`)  
Returns the value for this detail for one given node.

## Parameters

- `ignoreAged`: If no value should be returned which is older than the last request (not useful for nearly all cases).

**bool isVisible()**  
If this detail shall be a visible column in the view.

**QVariant requestValue**(`std::shared_ptr<FilesystemNode> node`, `sh::filesystem::Operation *op = 0`,  
`bool withNodeValues = false`, `bool withOperationsCache = true`)  
Requests to determine the value for this detail for one given node. Blocks until the value is available. Get it with the value method afterwards.

**bool sort\_doTypediff()**  
If sorting should separate files and directories.

**int sort\_order**(`std::shared_ptr<FilesystemNode> n1`, `std::shared_ptr<FilesystemNode> n2`)  
The sorting order.

**uint displayIndex()**  
Controls which place this column should get in the user interface.

**QVariant computeValue**(`std::shared_ptr<FilesystemNode> node`, `sh::filesystem::Operation *op`)  
Returns a freshly determined value for this column and the given node. It neither asks nor populates any caches or storages.

**int defaultWidth()**

**bool isRightAligned()**

**void applyValueByEurl**(`std::shared_ptr<const sh::filesystem::Eurl> eurl`,  
`sh::filesystem::Operation *op`, `QVariant value`)  
Applies the detail value to a given eurl (without using any caches).  
Used for transferring some details when a file transfer occurs.

Optionally implement the one of the `applyValueBy...` methods and register the instance with `sh::filesystem::FilesystemOperation::addTransferrableDetailColumn`. For performance reasons, you should decide to implement `applyValueByEurl`.

void **applyValueByNode** (std::shared\_ptr<FilesystemNode> *node*, *sh::filesystem::Operation* \**op*,  
QVariant *value*)

Applies the detail value to a given node (without using any caches).

Used for transferring some details when a file transfer occurs.

Optionally implement the one of the `applyValueBy...` methods and register the instance with `sh::filesystem::FilesystemOperation::addTransferrableDetailColumn`. For performance reasons, you should decide to implement `applyValueByEurl`.

void **doShutdown** ()

Executes singleton shutdown.

void **shutdown** ()

Shutdown down this singleton.

bool **isAlive** ()

Returns if this singleton is alive (true until its shutdown begins).

## Public Static Functions

QVariant **VALUE\_UNAVAILABLE** ()

A special value expressing that the value is not yet available.

void **registerKnownDetailColumn** (QString *name*, std::shared\_ptr<DetailColumn> *column*)

std::shared\_ptr<DetailColumn> **findKnownDetailColumn** (QString *name*)

## Public Static Attributes

const uint **DISPLAYINDEX\_CORE** = 10000

const uint **DISPLAYINDEX\_INTERESTING** = 20000

const uint **DISPLAYINDEX\_EXOTIC** = 30000

## Private Functions

**DetailColumnFilesize** ()

### 10.1.125 Class `sh::detailcolumns::DetailColumnMimetype`

**class** `sh::detailcolumns::DetailColumnMimetype` : **public** `sh::filesystem::DetailColumn`, **public** `sh::base::Singleton`

Detail column which determines the file's mimetype.

## Public Functions

QString **name** ()

The internal unique name.

QString **displayName** ()

The display name.

bool **isValueAvailable** (std::shared\_ptr<FilesystemNode> *node*)

Checks if a value for this detail is available for one given node.

QVariant **value** (std::shared\_ptr<FilesystemNode> *node*, bool *ignoreAged* = false)

Returns the value for this detail for one given node.

### Parameters

- *ignoreAged*: If no value should be returned which is older than the last request (not useful for nearly all cases).

bool **isVisible** ()

If this detail shall be a visible column in the view.

QVariant **requestValue** (std::shared\_ptr<FilesystemNode> *node*, *sh::filesystem::Operation* \**op* = 0, bool *withNodeValues* = false, bool *withOperationsCache* = true)

Requests to determine the value for this detail for one given node. Blocks until the value is available. Get it with the value method afterwards.

bool **sort\_doTypediff** ()

If sorting should separate files and directories.

int **sort\_order** (std::shared\_ptr<FilesystemNode> *n1*, std::shared\_ptr<FilesystemNode> *n2*)

The sorting order.

uint **displayIndex** ()

Controls which place this column should get in the user interface.

QVariant **computeValue** (std::shared\_ptr<FilesystemNode> *node*, *sh::filesystem::Operation* \**op*)

Returns a freshly determined value for this column and the given node. It neither asks nor populates any caches or storages.

int **defaultWidth** ()

bool **isRightAligned** ()

void **applyValueByEurl** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*, *sh::filesystem::Operation* \**op*, QVariant *value*)

Applies the detail value to a given eurl (without using any caches).

Used for transferring some details when a file transfer occurs.

Optionally implement the one of the `applyValueBy...` methods and register the instance with *sh::filesystem::FilesystemOperation::addTransferrableDetailColumn*. For performance reasons, you should decide to implement `applyValueByEurl`.

void **applyValueByNode** (std::shared\_ptr<FilesystemNode> *node*, *sh::filesystem::Operation* \**op*, QVariant *value*)

Applies the detail value to a given node (without using any caches).

Used for transferring some details when a file transfer occurs.

Optionally implement the one of the `applyValueBy...` methods and register the instance with *sh::filesystem::FilesystemOperation::addTransferrableDetailColumn*. For performance reasons, you should decide to implement `applyValueByEurl`.

void **doShutdown** ()  
 Executes singleton shutdown.

void **shutdown** ()  
 Shutdown down this singleton.

bool **isAlive** ()  
 Returns if this singleton is alive (true until its shutdown begins).

### Public Static Functions

QVariant **VALUE\_UNAVAILABLE** ()  
 A special value expressing that the value is not yet available.

void **registerKnownDetailColumn** (QString *name*, std::shared\_ptr<DetailColumn> *column*)

std::shared\_ptr<DetailColumn> **findKnownDetailColumn** (QString *name*)

### Public Static Attributes

const uint **DISPLAYINDEX\_CORE** = 10000

const uint **DISPLAYINDEX\_INTERESTING** = 20000

const uint **DISPLAYINDEX\_EXOTIC** = 30000

### Private Functions

DetailColumnMimetype ()

## 10.1.126 Class `sh::detailcolumns::DetailColumnMtime`

**class** `sh::detailcolumns::DetailColumnMtime` : **public** `sh::filesystem::DetailColumn`, **public** `sh::base::Singleton`  
 Detail column which determines the file's modification time.

### Public Functions

void **applyValueByEurl** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*,  
*sh::filesystem::Operation* \**op*, QVariant *value*) **override**

QString **name** ()  
 The internal unique name.

QString **displayName** ()  
 The display name.

bool **isValueAvailable** (std::shared\_ptr<FilesystemNode> *node*)  
 Checks if a value for this detail is available for one given node.

QVariant **value** (std::shared\_ptr<FilesystemNode> *node*, bool *ignoreAged* = false)  
 Returns the value for this detail for one given node.

### Parameters

- *ignoreAged*: If no value should be returned which is older than the last request (not useful for nearly all cases).

bool **isVisible** ()

If this detail shall be a visible column in the view.

QVariant **requestValue** (std::shared\_ptr<FilesystemNode> *node*, *sh::filesystem::Operation* \**op* = 0,  
bool *withNodeValues* = false, bool *withOperationsCache* = true)

Requests to determine the value for this detail for one given node. Blocks until the value is available. Get it with the value method afterwards.

bool **sort\_doTypediff** ()

If sorting should separate files and directories.

int **sort\_order** (std::shared\_ptr<FilesystemNode> *n1*, std::shared\_ptr<FilesystemNode> *n2*)

The sorting order.

uint **displayIndex** ()

Controls which place this column should get in the user interface.

QVariant **computeValue** (std::shared\_ptr<FilesystemNode> *node*, *sh::filesystem::Operation* \**op*)

Returns a freshly determined value for this column and the given node. It neither asks nor populates any caches or storages.

int **defaultWidth** ()

bool **isRightAligned** ()

void **applyValueByEurl** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*,  
*sh::filesystem::Operation* \**op*, QVariant *value*)

Applies the detail value to a given eurl (without using any caches).

Used for transferring some details when a file transfer occurs.

Optionally implement the one of the applyValueBy... methods and register the instance with *sh::filesystem::FilesystemOperation::addTransferrableDetailColumn*. For performance reasons, you should decide to implement applyValueByEurl.

void **applyValueByNode** (std::shared\_ptr<FilesystemNode> *node*, *sh::filesystem::Operation* \**op*,  
QVariant *value*)

Applies the detail value to a given node (without using any caches).

Used for transferring some details when a file transfer occurs.

Optionally implement the one of the applyValueBy... methods and register the instance with *sh::filesystem::FilesystemOperation::addTransferrableDetailColumn*. For performance reasons, you should decide to implement applyValueByEurl.

void **doShutdown** ()

Executes singleton shutdown.

void **shutdown** ()

Shutdown down this singleton.

bool **isAlive** ()

Returns if this singleton is alive (true until its shutdown begins).

## Public Static Functions

QVariant **VALUE\_UNAVAILABLE** ()

A special value expressing that the value is not yet available.

void **registerKnownDetailColumn** (QString *name*, std::shared\_ptr<DetailColumn> *column*)

std::shared\_ptr<DetailColumn> **findKnownDetailColumn** (QString *name*)

## Public Static Attributes

const uint **DISPLAYINDEX\_CORE** = 10000

const uint **DISPLAYINDEX\_INTERESTING** = 20000

const uint **DISPLAYINDEX\_EXOTIC** = 30000

## Private Functions

**DetailColumnMtime** ()

### 10.1.127 Class sh::detailcolumns::DetailColumnResolvedSymlink

**class** *sh::detailcolumns::DetailColumnResolvedSymlink* : **public** *sh::filesystem::DetailColumn*, **public** *sh::ba*

Detail column which determines the symlink target (by resolving links recursively).

This detail column has some active behavior. It informs the filesystem model about the ‘buddy nodes’.

## Public Functions

QString **name** ()

The internal unique name.

QString **displayName** ()

The display name.

bool **isValueAvailable** (std::shared\_ptr<FilesystemNode> *node*)

Checks if a value for this detail is available for one given node.

QVariant **value** (std::shared\_ptr<FilesystemNode> *node*, bool *ignoreAged* = false)

Returns the value for this detail for one given node.

## Parameters

- *ignoreAged*: If no value should be returned which is older than the last request (not useful for nearly all cases).

QVariant **requestValue** (std::shared\_ptr<FilesystemNode> *node*, *sh::filesystem::Operation* \**op* = 0, bool *withNodeValues* = false, bool *withOperationsCache* = true)

Requests to determine the value for this detail for one given node. Blocks until the value is available. Get it with the value method afterwards.

bool **sort\_doTypediff** ()

If sorting should separate files and directories.

int **sort\_order** (std::shared\_ptr<FilesystemNode> *n1*, std::shared\_ptr<FilesystemNode> *n2*)

The sorting order.

uint **displayIndex** ()

Controls which place this column should get in the user interface.

QVariant **computeValue** (std::shared\_ptr<FilesystemNode> *node*, *sh::filesystem::Operation* \**op*)

Returns a freshly determined value for this column and the given node. It neither asks nor populates any caches or storages.

int **defaultWidth** ()

bool **isRightAligned** ()

void **applyValueByEurl** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*,  
*sh::filesystem::Operation* \**op*, QVariant *value*)

Applies the detail value to a given eurl (without using any caches).

Used for transferring some details when a file transfer occurs.

Optionally implement the one of the `applyValueBy...` methods and register the instance with *sh::filesystem::FilesystemOperation::addTransferrableDetailColumn*. For performance reasons, you should decide to implement `applyValueByEurl`.

void **applyValueByNode** (std::shared\_ptr<FilesystemNode> *node*, *sh::filesystem::Operation* \**op*,  
QVariant *value*)

Applies the detail value to a given node (without using any caches).

Used for transferring some details when a file transfer occurs.

Optionally implement the one of the `applyValueBy...` methods and register the instance with *sh::filesystem::FilesystemOperation::addTransferrableDetailColumn*. For performance reasons, you should decide to implement `applyValueByEurl`.

void **doShutdown** ()

Executes singleton shutdown.

void **shutdown** ()

Shutdown down this singleton.

bool **isAlive** ()

Returns if this singleton is alive (true until its shutdown begins).

## Public Static Functions

QVariant **VALUE\_UNAVAILABLE** ()

A special value expressing that the value is not yet available.

void **registerKnownDetailColumn** (QString *name*, std::shared\_ptr<DetailColumn> *column*)

std::shared\_ptr<DetailColumn> **findKnownDetailColumn** (QString *name*)

## Public Static Attributes

const uint **DISPLAYINDEX\_CORE** = 10000

const uint **DISPLAYINDEX\_INTERESTING** = 20000

const uint **DISPLAYINDEX\_EXOTIC** = 30000



## Private Functions

`DetailColumnResolvedSymlink()`

### 10.1.128 Class `sh::exceptions::ArgumentException`

**class** `sh::exceptions::ArgumentException` : **public** `sh::exceptions::ProgramException`

Shallot exception for failed operation due to invalid arguments given to some program part. It typically allows resume but not retry (special cases may override each of them).

Subclassed by `sh::filesystem::EurlMisformattedException`

## Public Functions

**ArgumentException** (*ExceptionData* data)

QString **message** () **const**

QString **name** () **const**

QString **classes** () **const**

QString **details** () **const**

QString **callstack** () **const**

QString **auxiliary** () **const**

QString **customValue** (QString key) **const**

bool **isRuntimeException** () **const**

bool **isProgramException** () **const**

bool **isRetryable** () **const**

bool **isResumeable** () **const**

bool **isDetailsAreInteresting** () **const**

int **autoRetryRecommendedIn** () **const**

void **setResumeable** (bool v)

void **setReTryable** (bool v)

void **setCustomValue** (QString key, QString value)

bool **isClass** (QString classname)

*sh::exceptions::ExceptionData* **data** ()

## Public Static Functions

```
template<class Handler>
std::shared_ptr<RegisterHandler<Handler>> createRegisterHandler (Handler handler, QS-
                                                                    tack<Handler> *stack)
```

```
void executeGuarded (std::function<void>
    > fct int flags = 0 Executes some code with some standard exection handling around.
```

### Parameters

- *fct*: The code to execute guarded.
- *flags*: Flags of *ExecuteGuardFlag* for choosing the behavior.

```
void executeGuarded_errorpanel (std::function<void>
    > fct int flags = 0
```

```
QString _demangle (QString l)
```

```
void exceptionDialog (sh::exceptions::Exception &e, bool *doRetry)
```

## Public Static Attributes

```
const QString UNDEFINED_ERROR_OCCURRED = QObject::tr("An unspecified error occurred.")
```

```
const QString SHALLOT_MUST_CLOSE_TEXT = QObject::tr("Your Shallot process is disturbed by an error and needs to be c
```

```
const QString Value_isShallotException = "_isShallotException"
```

### 10.1.129 Class *sh::exceptions::CancelException*

```
class sh::exceptions::CancelException
```

A special exception for cancellation of some action on user behalf.

A very special exception outside of the hierarchy. It is only used by the infrastructure; never throw it directly.

### Public Functions

```
CancelException ()
```

### 10.1.130 Class *sh::exceptions::Exception*

```
class sh::exceptions::Exception
```

Shallot exception base class. Also contains some static methods for general work with exceptions.

Subclassed by *sh::exceptions::ProgramException*, *sh::exceptions::RuntimeException*,  
*sh::scripting::ScriptingEngine::ScriptedException*

## Public Functions

```

QString message () const
QString name () const
QString classes () const
QString details () const
QString callstack () const
QString auxiliary () const
QString customValue (QString key) const
bool isRuntimeException () const
bool isProgramException () const
bool isRetryable () const
bool isResumeable () const
bool isDetailsAreInteresting () const
int autoRetryRecommendedIn () const
void setResumeable (bool v)
void setReTryable (bool v)
void setCustomValue (QString key, QString value)
bool isClass (QString classname)
Exception (ExceptionData data)
Exception ()
sh::exceptions::ExceptionData data ()

```

## Public Static Functions

```

template<class Handler>
std::shared_ptr<RegisterHandler<Handler>> createRegisterHandler (Handler handler, QS-
                                                                    tack<Handler> *stack)

```

```

void executeGuarded (std::function<void>
    > fctint flags = 0) Executes some code with some standard exection handling around.

```

### Parameters

- fct: The code to execute guarded.
- flags: Flags of *ExecuteGuardFlag* for choosing the behavior.

```

void executeGuarded_errorpanel (std::function<void>
    > fctint flags = 0)
QString _demangle (QString l)
void exceptionDialog (sh::exceptions::Exception &e, bool *doRetry)

```

### Public Static Attributes

```
const QString UNDEFINED_ERROR_OCCURRED = QObject::tr("An unspecified error occurred.")
const QString SHALLOT_MUST_CLOSE_TEXT = QObject::tr("Your Shallot process is disturbed by an error and needs to be c
const QString Value_isShallotException = "_isShallotException"
```

### Private Static Functions

```
void exceptionDialog(QString error1, QString error2, QString details, QString icon, bool
                    mayRetry, bool mayClose, bool mayCancel, bool showLoglabelAndDetails,
                    bool *doRetry)
```

### Private Static Attributes

```
bool _inShutdown = false
QMutex _inShutdownMutex
class HandlerSettings
```

### Public Members

```
QString cancelText
QStack<std::function<bool (sh::exceptions::Exception&)>> exceptionHandler_retryable
QStack<std::function<void (sh::exceptions::Exception&)>> exceptionHandler_resumeable
QStack<std::function<void (sh::exceptions::Exception&)>> exceptionHandler_hard
QStack<std::function<void ()>> exceptionHandler_cancel
```

### Private Functions

```
HandlerSettings()
```

### Friends

```
friend class ExceptionHandlerSettingsManager
template<class Handler>
class RegisterHandler
```

**Public Functions**

**RegisterHandler** (*Handler handler*, QStack<*Handler*> \**stack*)

**~RegisterHandler** ()

**Private Members**

QStack<*Handler*> \*\_**stack**

**10.1.131 Class sh::exceptions::ExceptionData**

**class** *sh::exceptions::ExceptionData* : **public** QMap<QString, QString>

Used for specifying metadata for a *sh::exceptions::Exception*.

**Public Functions**

**ExceptionData** (**const** *ExceptionData* &*o*)

**ExceptionData** (**const** QMap<QString, QString> &*o*)

**ExceptionData** ()

*ExceptionData* **name** (QString *name*)

*ExceptionData* **classes** (QString *classes*)

*ExceptionData* **aux** (QString *aux*)

*ExceptionData* **details** (QString *details*)

*ExceptionData* **message** (QString *message*)

*ExceptionData* **runtime** ()

*ExceptionData* **program** ()

*ExceptionData* **retryable** (bool *v* = true)

*ExceptionData* **resumeable** (bool *v* = true)

*ExceptionData* **autoRetryRecommended** (int *v* = -1)

*ExceptionData* **detailsAreInteresting** (bool *v* = true)

**10.1.132 Class sh::exceptions::ExceptionHandlerSettingsManager**

**class** *sh::exceptions::ExceptionHandlerSettingsManager* : **public** *sh::base::Singleton*

Managing how to default-handle unhandled exceptions.

### Public Functions

*Exception::HandlerSettings* \***handlerSettings** ()  
Returns the current *Exception::HandlerSettings*.

**~ExceptionHandlerSettingsManager** ()

void **doInitialize** ()  
Executes singleton initialization.

void **doShutdown** ()  
Executes singleton shutdown.

void **shutdown** ()  
Shutdown down this singleton.

bool **isAlive** ()  
Returns if this singleton is alive (true until its shutdown begins).

### Private Functions

**ExceptionHandlerSettingsManager** ()

### Private Members

QMap<QThread\*, *Exception::HandlerSettings*\*> **\_handlerSettings**  
QMutex **\_mutex**

## 10.1.133 Class sh::exceptions::Exception::HandlerSettings

**class** *sh::exceptions::Exception::HandlerSettings*

### Public Members

QString **cancelText**

QStack<std::function<bool (*sh::exceptions::Exception*&) >> **exceptionHandler\_retryable**

QStack<std::function<void (*sh::exceptions::Exception*&) >> **exceptionHandler\_resumeable**

QStack<std::function<void (*sh::exceptions::Exception*&) >> **exceptionHandler\_hard**

QStack<std::function<void () >> **exceptionHandler\_cancel**

### Private Functions

**HandlerSettings** ()

## Friends

**friend class** ExceptionHandlerSettingsManager

### 10.1.134 Class `sh::exceptions::Exception::RegisterHandler`

```
template<class Handler>
class sh::exceptions::Exception::RegisterHandler
```

#### Public Functions

```
RegisterHandler (Handler handler, QStack<Handler> *stack)
~RegisterHandler ()
```

#### Private Members

```
QStack<Handler> *_stack
```

### 10.1.135 Class `sh::exceptions::IOException`

```
class sh::exceptions::IOException : public sh::exceptions::RuntimeException
```

Shallot exception in IO. It allows resume and typically allows retry (special cases may override each of them).

Subclassed by `sh::exceptions::PermissionDeniedException`, `sh::filesystem::Operation::MaxAllowedSizeRatioPerPartExceededEx`

#### Public Functions

```
IOException (ExceptionData data)
QString message () const
QString name () const
QString classes () const
QString details () const
QString callstack () const
QString auxiliary () const
QString customValue (QString key) const
bool isRuntimeException () const
bool isProgramException () const
bool isRetryable () const
bool isResumeable () const
bool isDetailsAreInteresting () const
int autoRetryRecommendedIn () const
void setResumeable (bool v)
void setReTryable (bool v)
```

```
void setCustomValue (QString key, QString value)
```

```
bool isClass (QString classname)
```

```
sh::exceptions::ExceptionData data ()
```

### Public Static Functions

```
template<class Handler>
```

```
std::shared_ptr<RegisterHandler<Handler>> createRegisterHandler (Handler handler, QS-  
tack<Handler> *stack)
```

```
void executeGuarded (std::function<void>)
```

```
> fcntl flags = 0 Executes some code with some standard execution handling around.
```

#### Parameters

- *fcntl*: The code to execute guarded.
- *flags*: Flags of *ExecuteGuardFlag* for choosing the behavior.

```
void executeGuarded_errorpanel (std::function<void>)
```

```
> fcntl flags = 0
```

```
QString _demangle (QString l)
```

```
void exceptionDialog (sh::exceptions::Exception &e, bool *doRetry)
```

### Public Static Attributes

```
const QString UNDEFINED_ERROR_OCCURRED = QObject::tr("An unspecified error occurred.")
```

```
const QString SHALLOT_MUST_CLOSE_TEXT = QObject::tr("Your Shallot process is disturbed by an error and needs to be c
```

```
const QString Value_isShallotException = "_isShallotException"
```

## 10.1.136 Class *sh::exceptions::PermissionDeniedException*

```
class sh::exceptions::PermissionDeniedException : public sh::exceptions::IOException
```

Shallot exception for something forbidden was tried to execute. It allows resume and retry (special cases may override each of them).

### Public Functions

```
PermissionDeniedException (ExceptionData data)
```

```
QString message () const
```

```
QString name () const
```

```
QString classes () const
```

```
QString details () const
```

```
QString callstack () const
```

```
QString auxiliary () const
```

```
QString customValue (QString key) const
```



```

bool isRuntimeException () const
bool isProgramException () const
bool isRetryable () const
bool isResumeable () const
bool isDetailsAreInteresting () const
int autoRetryRecommendedIn () const
void setResumeable (bool v)
void setReTryable (bool v)
void setCustomValue (QString key, QString value)
bool isClass (QString classname)
sh::exceptions::ExceptionData data ()

```

## Public Static Functions

```

template<class Handler>
std::shared_ptr<RegisterHandler<Handler>> createRegisterHandler (Handler handler, QS-
                                                                    tack<Handler> *stack)

void executeGuarded (std::function<void>)
    > fcntl flags = 0 Executes some code with some standard exection handling around.

```

### Parameters

- *fct*: The code to execute guarded.
- *flags*: Flags of *ExecuteGuardFlag* for choosing the behavior.

```

void executeGuarded_errorpanel (std::function<void>)
    > fcntl flags = 0

QString _demangle (QString l)

void exceptionDialog (sh::exceptions::Exception &e, bool *doRetry)

```

## Public Static Attributes

```

const QString UNDEFINED_ERROR_OCCURRED = QObject::tr("An unspecified error occurred.")
const QString SHALLOT_MUST_CLOSE_TEXT = QObject::tr("Your Shallot process is disturbed by an error and needs to be c
const QString Value_isShallotException = "_isShallotException"

```

### 10.1.137 Class `sh::exceptions::ProgramException`

**class** `sh::exceptions::ProgramException` : **public** `sh::exceptions::Exception`

Shallot exception for internal bugs in Shallot. It optionally allows resume but no retry (special cases may override each of them).

Subclassed by `sh::exceptions::ArgumentException`, `sh::exceptions::ThreadingException`

#### Public Functions

**ProgramException** (*ExceptionData* data)

QString **message** () **const**

QString **name** () **const**

QString **classes** () **const**

QString **details** () **const**

QString **callstack** () **const**

QString **auxiliary** () **const**

QString **customValue** (QString key) **const**

bool **isRuntimeException** () **const**

bool **isProgramException** () **const**

bool **isRetryable** () **const**

bool **isResumeable** () **const**

bool **isDetailsAreInteresting** () **const**

int **autoRetryRecommendedIn** () **const**

void **setResumeable** (bool v)

void **setReTryable** (bool v)

void **setCustomValue** (QString key, QString value)

bool **isClass** (QString classname)

*sh::exceptions::ExceptionData* **data** ()

#### Public Static Functions

template<class **Handler**>

std::shared\_ptr<RegisterHandler<*Handler*>> **createRegisterHandler** (*Handler* handler, QStack<*Handler*> \*stack)

void **executeGuarded** (std::function<void>)

> *fct* int *flags* = 0 Executes some code with some standard execution handling around.

#### Parameters

- *fct*: The code to execute guarded.
- *flags*: Flags of *ExecuteGuardFlag* for choosing the behavior.

```
void executeGuarded_errorpanel (std::function<void>
    > fcntl_flags = 0)
QString _demangle (QString l)
void exceptionDialog (sh::exceptions::Exception &e, bool *doRetry)
```

### Public Static Attributes

```
const QString UNDEFINED_ERROR_OCCURRED = QObject::tr("An unspecified error occurred.")
const QString SHALLOT_MUST_CLOSE_TEXT = QObject::tr("Your Shallot process is disturbed by an error and needs to be c
const QString Value_isShallotException = "_isShallotException"
```

## 10.1.138 Class *sh::exceptions::RuntimeException*

```
class sh::exceptions::RuntimeException : public sh::exceptions::Exception
```

Shallot exception for failed operation due to (often external) runtime effects. It allows resume and optionally allows retry (special cases may override each of them).

Subclassed by *sh::exceptions::IOException*

### Public Functions

```
RuntimeException (ExceptionData data)
QString message () const
QString name () const
QString classes () const
QString details () const
QString callstack () const
QString auxiliary () const
QString customValue (QString key) const
bool isRuntimeException () const
bool isProgramException () const
bool isRetryable () const
bool isResumeable () const
bool isDetailsAreInteresting () const
int autoRetryRecommendedIn () const
void setResumeable (bool v)
void setReTryable (bool v)
void setCustomValue (QString key, QString value)
bool isClass (QString classname)
sh::exceptions::ExceptionData data ()
```

## Public Static Functions

```
template<class Handler>
std::shared_ptr<RegisterHandler<Handler>> createRegisterHandler (Handler handler, QS-
                                                                    tack<Handler> *stack)
```

```
void executeGuarded (std::function<void>
    > fct int flags = 0 Executes some code with some standard exection handling around.
```

### Parameters

- *fct*: The code to execute guarded.
- *flags*: Flags of *ExecuteGuardFlag* for choosing the behavior.

```
void executeGuarded_errorpanel (std::function<void>
    > fct int flags = 0
```

```
QString _demangle (QString l)
```

```
void exceptionDialog (sh::exceptions::Exception &e, bool *doRetry)
```

## Public Static Attributes

```
const QString UNDEFINED_ERROR_OCCURRED = QObject::tr("An unspecified error occurred.")
```

```
const QString SHALLOT_MUST_CLOSE_TEXT = QObject::tr("Your Shallot process is disturbed by an error and needs to be c
```

```
const QString Value_isShallotException = "_isShallotException"
```

### 10.1.139 Class *sh::exceptions::ThreadAbortException*

```
class sh::exceptions::ThreadAbortException
```

Only used by the infrastructure; never throw it directly.

### Public Functions

```
ThreadAbortException ()
```

### 10.1.140 Class *sh::exceptions::ThreadingException*

```
class sh::exceptions::ThreadingException : public sh::exceptions::ProgramException
```

Shallot exception for misuse of threading (e.g. wrong caller thread). It does not allow resume (special cases may override each of them).

## Public Functions

```

ThreadingException (ExceptionData data)
QString message () const
QString name () const
QString classes () const
QString details () const
QString callstack () const
QString auxiliary () const
QString customValue (QString key) const
bool isRuntimeException () const
bool isProgramException () const
bool isRetryable () const
bool isResumeable () const
bool isDetailsAreInteresting () const
int autoRetryRecommendedIn () const
void setResumeable (bool v)
void setReTryable (bool v)
void setCustomValue (QString key, QString value)
bool isClass (QString classname)
sh::exceptions::ExceptionData data ()

```

## Public Static Functions

```

template<class Handler>
std::shared_ptr<RegisterHandler<Handler>> createRegisterHandler (Handler handler, QS-
                                                                    tack<Handler> *stack)

```

```

void executeGuarded (std::function<void>)
    > fctint flags = 0 Executes some code with some standard exection handling around.

```

### Parameters

- *fct*: The code to execute guarded.
- *flags*: Flags of *ExecuteGuardFlag* for choosing the behavior.

```

void executeGuarded_errorpanel (std::function<void>)
    > fctint flags = 0
QString _demangle (QString l)
void exceptionDialog (sh::exceptions::Exception &e, bool *doRetry)

```

### Public Static Attributes

**const** QString **UNDEFINED\_ERROR\_OCCURRED** = QObject::tr("An unspecified error occurred.")

**const** QString **SHALLOT\_MUST\_CLOSE\_TEXT** = QObject::tr("Your Shallot process is disturbed by an error and needs to be c

**const** QString **Value\_isShallotException** = "\_isShallotException"

## 10.1.141 Class `sh::filepropertydialogtabs::FilePropertyDialogTabExtendedAttributes`

**class** `sh::filepropertydialogtabs::FilePropertyDialogTabExtendedAttributes` : **public** `sh::ui::FilePro`  
Properties dialog tab for extended attributes.

### Public Functions

**FilePropertyDialogTabExtendedAttributes** ()

QString **title** () **override**

The tab title. .

QList<QString> **properties** () **override**

Returns the list of property labels (one entry for each widget).

Each property is displayed with a header and a widget (created in `createWidget`) with some content (populated in `updateWidget`).

void **populateWidget** (int *i*, `sh::ui::FilePropertyDialogTabActionsView` \**widget*) **override**

Populates the tab widget for the *i*-th property.

Typically uses the `FilePropertyDialog::create*` methods to create sub-widgets.

See also `dialog()`.

void **updateWidget** (int *i*, `sh::ui::FilePropertyDialogTabActionsView` \**widget*,  
`sh::filesystem::Operation` \**op*) **override**

Populates the widget for the *i*-th property with actual content. .

QString **titleWithoutMnemonic** ()

The tab title without mnemonic.

void **refresh** ()

Refreshes the complete content.

Typically called after some user actions in a property widget require that all the other content must be refreshed.

QList<std::shared\_ptr<`sh::filesystem::FilesystemNode`>> **nodes** ()

The nodes to show.

`FilePropertyDialogTabActionsView` \***widgetAt** (int *i*)

Returns the widget for the *i*-th property.

int **widgetCount** ()

Returns the number of widgets.

std::shared\_ptr<`FilePropertyDialog`> **dialog** ()

Returns the associated dialog.

## Public Static Functions

void **doInitialize** ()

void **doShutdown** ()

## 10.1.142 Class `sh::filepropertydialogtabs::FilePropertyDialogTabExtendedAttributes::ActionAddAttribute`

**class** `sh::filepropertydialogtabs::FilePropertyDialogTabExtendedAttributes::ActionAddAttribute`

## Public Functions

**ActionAddAttribute** (std::shared\_ptr<`sh::filepropertydialogtabs::FilePropertyDialogTabExtendedAttributes`> *tab*)

void **execute** ()  
Executes this action.

void **execute** (`sh::actions::ActionExecutionInfo` \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<AbstractActionItem> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<AbstractActionItem> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

### 10.1.143 Class `sh::filepropertydialogtabs::FilePropertyDialogTabExtendedAttributes::ActionCh`

**class** `sh::filepropertydialogtabs::FilePropertyDialogTabExtendedAttributes::ActionChangeAtt`

#### Public Functions

**ActionChangeAttribute** (std::shared\_ptr<`sh::filepropertydialogtabs::FilePropertyDialogTabExtendedAttributes`>  
*tab*)

void **execute** ()  
Executes this action.

void **execute** (`sh::actions::ActionExecutionInfo` \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.



QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<AbstractActionItem> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<AbstractActionItem> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

### 10.1.144 Class `sh::filepropertydialogtabs::FilePropertyDialogTabExtendedAttributes::ActionRemoveAttribute`

`class sh::filepropertydialogtabs::FilePropertyDialogTabExtendedAttributes::ActionRemoveAttribute`

#### Public Functions

**ActionRemoveAttribute** (std::shared\_ptr<`sh::filepropertydialogtabs::FilePropertyDialogTabExtendedAttributes`>  
*tab*)

void **execute** ()  
Executes this action.

void **execute** (`sh::actions::ActionExecutionInfo` \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<AbstractActionItem> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<AbstractActionItem> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0)  
Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

### 10.1.145 Class sh::filepropertydialogtabs::FilePropertyDialogTabGeneral

**class** *sh::filepropertydialogtabs::FilePropertyDialogTabGeneral* : public *sh::ui::FilePropertyDialogTab*  
Properties dialog tab for general infos.

#### Public Functions

**FilePropertyDialogTabGeneral** ()

QString **title** () **override**  
The tab title. .

QList<QString> **properties** () **override**  
Returns the list of property labels (one entry for each widget).  
  
Each property is displayed with a header and a widget (created in `createWidget`) with some content (populated in `updateWidget`).

void **populateWidget** (int *i*, *sh::ui::FilePropertyDialogTabActionsView* \**widget*) **override**  
Populates the tab widget for the *i*-th property.  
  
Typically uses the `FilePropertyDialog::create*` methods to create sub-widgets.  
  
See also *dialog()*.

```
void updateWidget (int i, sh::ui::FilePropertyDialogTabActionsView *widget,
                  sh::filesystem::Operation *op) override
    Populates the widget for the i-th property with actual content. .

QString titleWithoutMnemonic ()
    The tab title without mnemonic.

void refresh ()
    Refreshes the complete content.

    Typically called after some user actions in a property widget require that all the other content must be
    refreshed.

QList<std::shared_ptr<sh::filesystem::FilesystemNode>> nodes ()
    The nodes to show.

FilePropertyDialogTabActionsView *widgetAt (int i)
    Returns the widget for the i-th property.

int widgetCount ()
    Returns the number of widgets.

std::shared_ptr<FilePropertyDialog> dialog ()
    Returns the associated dialog.
```

### Public Static Functions

```
void doInitialize ()

void doShutdown ()
```

## 10.1.146 Class sh::filepropertydialogtabs::FilePropertyDialogTabUnixPermissions

```
class sh::filepropertydialogtabs::FilePropertyDialogTabUnixPermissions : public sh::ui::FilePropertyDialogTab
    Properties dialog tab for unix permissions.
```

### Public Functions

```
FilePropertyDialogTabUnixPermissions ()

QString title () override
    The tab title. .

QList<QString> properties () override
    Returns the list of property labels (one entry for each widget).

    Each property is displayed with a header and a widget (created in createWidget) with some content (pop-
    ulated in updateWidget).

void populateWidget (int i, sh::ui::FilePropertyDialogTabActionsView *widget) override
    Populates the tab widget for the i-th property.

    Typically uses the FilePropertyDialog::create* methods to create sub-widgets.

    See also dialog().

void updateWidget (int i, sh::ui::FilePropertyDialogTabActionsView *widget,
                  sh::filesystem::Operation *op) override
    Populates the widget for the i-th property with actual content. .
```

QString **titleWithoutMnemonic** ()  
The tab title without mnemonic.

void **refresh** ()  
Refreshes the complete content.

Typically called after some user actions in a property widget require that all the other content must be refreshed.

QList<std::shared\_ptr<sh::filesystem::FilesystemNode>> **nodes** ()  
The nodes to show.

FilePropertyDialogTabActionsView \***widgetAt** (int *i*)  
Returns the widget for the *i*-th property.

int **widgetCount** ()  
Returns the number of widgets.

std::shared\_ptr<FilePropertyDialog> **dialog** ()  
Returns the associated dialog.

## Public Static Functions

QString **getUserName** (int *uid*)

QString **getGroupName** (int *gid*)

QMap<int, QString> **getAllUsers** ()

QMap<int, QString> **getAllGroups** ()

void **doInitialize** ()

void **doShutdown** ()

## Private Functions

void **slot\_buttontriggered** (int *i*)

QString **userPermissionsDescription** (int *flags*, int *rflag*, int *wflag*, int *xflag*)

**class ActionChange** : public *sh::actions::ActionActionItem*

## Public Functions

**ActionChange** (std::shared\_ptr<sh::filepropertydialogtabs::FilePropertyDialogTabUnixPermissions>  
*tab*)

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcut** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<AbstractActionItem> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<AbstractActionItem> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

### 10.1.147 Class `sh::filepropertydialogtabs::FilePropertyDialogTabUnixPermissions::ActionChange`

```
class sh::filepropertydialogtabs::FilePropertyDialogTabUnixPermissions::ActionChange : public
```

#### Public Functions

**ActionChange** (std::shared\_ptr<*sh::filepropertydialogtabs::FilePropertyDialogTabUnixPermissions*>  
*tab*)

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<AbstractActionItem> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **setParentAction** (std::shared\_ptr<AbstractActionItem> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

### 10.1.148 Class `sh::filepropertydialogtabs::FilePropertyDialogTabWindows`

**class** `sh::filepropertydialogtabs::FilePropertyDialogTabWindows` : **public** `sh::ui::FilePropertyDialogTab`  
Properties dialog tab for unix permissions.

#### Public Functions

QString **title** () = 0  
The tab title. .

QString **titleWithoutMnemonic** ()  
The tab title without mnemonic.

QList<QString> **properties** () = 0  
Returns the list of property labels (one entry for each widget).  
  
Each property is displayed with a header and a widget (created in `createWidget`) with some content (populated in `updateWidget`).

void **populateWidget** (int *i*, FilePropertyDialogTabActionsView \**widget*) = 0  
Populates the tab widget for the *i*-th property.  
  
Typically uses the `FilePropertyDialog::create*` methods to create sub-widgets.  
  
See also `dialog()`.



void **updateWidget** (int *i*, FilePropertyDialogTabActionsView \**widget*, *sh::filesystem::Operation* \**op*)  
     = 0  
     Populates the widget for the *i*-th property with actual content. .

void **refresh** ()  
     Refreshes the complete content.  
     Typically called after some user actions in a property widget require that all the other content must be refreshed.

QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> **nodes** ()  
     The nodes to show.

FilePropertyDialogTabActionsView \***widgetAt** (int *i*)  
     Returns the widget for the *i*-th property.

int **widgetCount** ()  
     Returns the number of widgets.

std::shared\_ptr<FilePropertyDialog> **dialog** ()  
     Returns the associated dialog.

### 10.1.149 Class *sh::filesystem::AdhocFilesystemNodeList*

**class** *sh::filesystem::AdhocFilesystemNodeList* : **public** *sh::filesystem::FilesystemNodeList*  
     This *FilesystemNodeList* subclass is used for spontaneously fetching a fresh list of node children.

Node additions and removals will not touch the model.

Node instances inside it will be deleted together with this list.

#### Public Functions

**AdhocFilesystemNodeList** ()  
     Is intended to be directly constructed from everywhere.

**~AdhocFilesystemNodeList** ()

void **addItems** (QSet<std::shared\_ptr<*FilesystemNode*>> *nodes*)

void **addItem** (std::shared\_ptr<*FilesystemNode*> *node*)

void **removeItems** (QSet<std::shared\_ptr<*FilesystemNode*>> *nodes*)

void **removeItem** (std::shared\_ptr<*FilesystemNode*> *node*)

void **resetItems** (*sh::filesystem::FilesystemNodeType* *type*, QSet<std::shared\_ptr<*FilesystemNode*>> *nodes*)

std::shared\_ptr<*FilesystemNode*> **myNode** ()  
     Returns the node which owns this children list. The result may be 0 for non model-backed lists.

void **addItems** (QSet<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> *nodes*) = 0  
     Adds items to this list. In a model-backed list, this triggers the internal model mounting calls. It is not allowed to call this method twice with the same node.

void **addItem** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> *node*) = 0  
     Adds an item to this list. In a model-backed list, this triggers the internal model mounting calls. It is not allowed to call this method twice with the same node.

void **removeItems** (QSet<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> *nodes*) = 0  
Removes nodes from this list. In a model-backed list, this triggers the internal model unmounting calls. It is not allowed to call this method with a node, which is not contained in that list.

void **removeItem** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> *node*) = 0  
Removes a node from this list. In a model-backed list, this triggers the internal model unmounting calls. It is not allowed to call this method with a node, which is not contained in that list.

void **resetItems** (*sh::filesystem::FilesystemNodeType* *type*, QSet<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> *nodes*) = 0  
Resets the list for a given node type to a given new list of children nodes.

bool **contains** (std::shared\_ptr<*FilesystemNode*> *node*)  
Checks if this list contains a given node.

const QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> **\*nodes** ()  
Returns the list of nodes currently stored in this instance.

### 10.1.150 Class *sh::filesystem::DetailColumn*

**class** *sh::filesystem::DetailColumn* : **public** std::enable\_shared\_from\_this<*DetailColumn*>  
Abstract base class for a detail column (on filesystem nodes).

Those can e.g. be seen in the file list view, but can also be queried internally by other places in code.

It encapsulates the retrieval logic and metadata for one piece of additional information a *sh::filesystem::FilesystemNode* can have (e.g. the filesize). Retrieving the values is designed to be asynchronous. Each instance represents one column, while the actual logic is implemented in subclasses. For a new detail column, subclass this class and implement at least `determineValue`.

Subclassed by *sh::detailcolumns::DetailColumnCustomAttributes*, *sh::detailcolumns::DetailColumnDirectSymlinkTarget*, *sh::detailcolumns::DetailColumnExtendedAttributes*, *sh::detailcolumns::DetailColumnFilesize*, *sh::detailcolumns::DetailColumnMimetype*, *sh::detailcolumns::DetailColumnMtime*, *sh::detailcolumns::DetailColumnResolvedSymlink*, *sh::filesystemhandlers::ShareFilesystemHandler::ArchivedSizeDetailColumn*, *sh::scripting::api::ApiDetailColumn*

#### Public Functions

QString **name** ()  
The internal unique name.

QString **displayName** ()  
The display name.

bool **isValueAvailable** (std::shared\_ptr<*FilesystemNode*> *node*)  
Checks if a value for this detail is available for one given node.

QVariant **value** (std::shared\_ptr<*FilesystemNode*> *node*, bool *ignoreAged* = false)  
Returns the value for this detail for one given node.

#### Parameters

- *ignoreAged*: If no value should be returned which is older than the last request (not useful for nearly all cases).

QString **displayValue** (std::shared\_ptr<*FilesystemNode*> *node*, *sh::filesystem::FilesystemModelFileviewProxy* *\*viewmodel*) **const**  
Returns the stringified value for this details for one given node considering the configuration of a view.

bool **isVisible** ()

If this detail shall be a visible column in the view.

QVariant **requestValue** (std::shared\_ptr<FilesystemNode> node, sh::filesystem::Operation \*op = 0,  
bool withNodeValues = false, bool withOperationsCache = true)

Requests to determine the value for this detail for one given node. Blocks until the value is available. Get it with the value method afterwards.

bool **sort\_doTypediff** ()

If sorting should separate files and directories.

int **sort\_order** (std::shared\_ptr<FilesystemNode> n1, std::shared\_ptr<FilesystemNode> n2)

The sorting order.

uint **displayIndex** ()

Controls which place this column should get in the user interface.

QVariant **computeValue** (std::shared\_ptr<FilesystemNode> node, sh::filesystem::Operation \*op)

Returns a freshly determined value for this column and the given node. It neither asks nor populates any caches or storages.

int **defaultWidth** ()

bool **isRightAligned** ()

~DetailColumn ()

void **applyValueByEurl** (std::shared\_ptr<const sh::filesystem::Eurl> eurl,  
sh::filesystem::Operation \*op, QVariant value)

Applies the detail value to a given eurl (without using any caches).

Used for transferring some details when a file transfer occurs.

Optionally implement the one of the applyValueBy... methods and register the instance with sh::filesystem::FilesystemOperation::addTransferrableDetailColumn. For performance reasons, you should decide to implement applyValueByEurl.

void **applyValueByNode** (std::shared\_ptr<FilesystemNode> node, sh::filesystem::Operation \*op,  
QVariant value)

Applies the detail value to a given node (without using any caches).

Used for transferring some details when a file transfer occurs.

Optionally implement the one of the applyValueBy... methods and register the instance with sh::filesystem::FilesystemOperation::addTransferrableDetailColumn. For performance reasons, you should decide to implement applyValueByEurl.

## Public Static Functions

QVariant **VALUE\_UNAVAILABLE** ()

A special value expressing that the value is not yet available.

void **registerKnownDetailColumn** (QString name, std::shared\_ptr<DetailColumn> column)

std::shared\_ptr<DetailColumn> **findKnownDetailColumn** (QString name)

### Public Static Attributes

```
const uint DISPLAYINDEX_CORE = 10000
const uint DISPLAYINDEX_INTERESTING = 20000
const uint DISPLAYINDEX_EXOTIC = 30000
```

### Private Members

```
QString _displayName
QString _name
uint _displayIndex
bool _sort_doTypediff
int _defaultWidth
bool _isRightAligned
```

### Private Static Attributes

```
QMap<QString, std::shared_ptr<DetailColumn>> _knownDetailColumns
QMutex _knownDetailColumnsMutex
```

### Friends

```
friend class sh::filesystem::FilesystemNode
```

## 10.1.151 Class sh::filesystem::Eurl

```
class sh::filesystem::Eurl : public std::enable_shared_from_this<Eurl>
    Filesystem paths.
```

A Extended Uniform Resource Locator is something like a [URL](#).

It is more general since it can also be nested. It is something like `zip:/{}/zippedfolder/zippedfile`.

You get [Eurl](#) instances only from the factory methods. There will never be two instances with the same textual value.

Please note: Instances can be associated with any kind of elements in the filesystem (files, directories, links, ...). It might also point to something which does not exist at all. The documentation sometimes explicitly makes a difference between those kinds (files, directories, links, ...; often called ‘node type’). But it often uses the term ‘file’ implicitly while meaning all kinds of elements; assuming that e.g. a directory is just a special kind of a file. It should be clear from the particular context which meaning applies.

## Public Functions

### QString **asString()** **const**

Returns the textual value. This is what *Eurl.fromString* would expect as parameter.

### QString **hostname()** **const**

Returns the hostname part (from the outer url of this *Eurl*). Examples: "livingroom-pc" for smb://livingroom-pc/foo/bar/baz. "" for .

### QString **path()** **const**

Returns the path part (from the outer url of this *Eurl*). Examples: "/foo/bar/baz" for smb://livingroom-pc/foo/bar/baz. "/" for . "/" for .

### QString **basename()** **const**

Returns the last path segment. This is the text behind the last slash. Examples: "baz" for . "" for .

### QString **scheme()** **const**

Returns the scheme (from the outer url of this shallot.Eurl). This is what comes before the ://. Example: "file" for .

### std::shared\_ptr<const *Eurl*> **outerUrl()** **const**

Returns a new *Eurl* containing only the outer part of this one (strips the embeddings). Example: foobar://host/foo/bar/baz for foobar://[zip://[]//d/e]//foo/bar/baz.

### std::shared\_ptr<const *Eurl*> **outermostInnerEurl()** **const**

Returns a new *Eurl* containing only the embedding of this one. Example: zip://[]//d/e for foobar://[zip://[]//d/e]//foo/bar/baz.

### std::shared\_ptr<const *Eurl*> **withAppendedSegment**(QString *basename*) **const**

Returns a new *Eurl* from this one with "/basename" appended. The parameter is assumed to be a single path segment.

### std::shared\_ptr<const *Eurl*> **withAppendedSegments**(QString *path*) **const**

Returns a new *Eurl* with path segments "/pa/t/h/" appended. The parameter may contain "/"'s for dividing path segments.

### std::shared\_ptr<const *Eurl*> **root()** **const**

Returns the root *Eurl* from this one. Example: zip://[]// for zip://[]//foo/bar/baz.

### std::shared\_ptr<const *Eurl*> **enwrapWithOuterUrl**(QString *scheme*, QString *hostname*, QString *path*) **const**

Returns a new *Eurl* containing this one packed as embedding and new outer parts scheme, hostname and path. Example: scheme://[]/hostname/p/a/t/h for .

### std::shared\_ptr<const *Eurl*> **parentSegment()** **const**

Returns the parent *Eurl*. At first, this traverses path segments. For a root path eurl with embeddings, it returns the embedding. If none are available, it returns 0. Examples: foo://host/foo for foo://host/foo/bar. foo://host/ for foo://host/boo. foo://[bar://host/goo]/anotherhost/ for foo://[bar://host/goo]/anotherhost/boo. bar://host/foo for foo://[bar://host/foo]/host/. 0 for foo://host/.

### bool **hasInnerUrls()** **const**

Checks if this *Eurl* has embeddings. Examples: true for foo://[bar:///foo]/host/. false for foo://host/.

### bool **outerUrlIsRootDirectory()** **const**

Checks if this *Eurl* is a root path (with or without embeddings). Examples: true for foo://host/. false for foo://host/a. true for foo://[bar:///goo]/host/. false for foo://[bar:///goo]/host/a. true for foo://[bar:///goo]//. false for foo://[bar:///goo]//a.

bool **hasParentSegment** () const

Checks if this *Eurl* has a parent segment. This indicates if *Eurl.parentSegment* would return 0.

bool **isPrefixOf** (std::shared\_ptr<const *Eurl*> longer) const

Checks if this *Eurl* is a prefix of another one. This is not an equivalent to a string comparison but it checks parent relationships according to *Eurl.parentSegment*.

**Eurl** (QString *eurlstring*)

Constructed only by the infrastructure and made available otherwise.

**~Eurl** ()

## Public Static Functions

std::shared\_ptr<const *Eurl*> **fromString** (QString *eurlstring*)

Constructs a new *Eurl* by string (what *Eurl.asString* would return).

std::shared\_ptr<const *Eurl*> **create** (QString *scheme*, QString *hostname*, QString *path*)

Constructs a new *Eurl* by scheme name, hostname and a path.

std::shared\_ptr<const *Eurl*> **create** (QString *scheme*, const *Eurl* \*inner, QString *hostname*,  
QString *path*)

Constructs a new *Eurl* by scheme name, an inner eurl, a hostname and a path.

void **filenameCheck** (QString *filename*)

Checks if a name is a valid filename. If not, *EurlMisformattedException* is thrown.

void **doInitialize** ()

void **doShutdown** ()

## Public Static Attributes

const QChar **WRAPPER\_BEGIN** = '['

The character marking the begin of an embedding.

const QChar **WRAPPER\_END** = ']'

The character marking the end of an embedding.

const QString **FORBIDDEN\_FILENAME\_CHARACTERS** = QString("/")

Characters which are forbidden in filenames.

## Private Members

const QString **\_eurlstring**

std::shared\_ptr<const *Eurl*> **\_cache\_parentsegment** = 0

std::shared\_ptr<const *Eurl*> **\_cache\_outerurl** = 0

bool **\_cache\_outerurl\_isthis** = false

std::shared\_ptr<const *Eurl*> **\_cache\_outermostinnereurl** = 0

std::shared\_ptr<const *Eurl*> **\_cache\_root** = 0

bool **\_cache\_root\_isthis** = false

QString **\_cache\_basename**

QString **\_cache\_hostname**

QString **\_cache\_path**  
 QString **\_cache\_scheme**

### Private Static Functions

QString **check\_scheme** (QString *scheme*)  
 QString **check\_inner** (QString *inner*)  
 QString **check\_hostname** (QString *hostname*)  
 QString **check\_path** (QString *path*)  
 QString **check\_filename** (QString *filename*)  
 QString **\_escape** (QString *s*)  
 QString **\_unescape** (QString *s*)  
 std::shared\_ptr<const *Eurl*> **createNOCHECK** (QString *scheme*, const *Eurl* \**inner*, QString *host-name*, QString *path*)  
 std::shared\_ptr<const *Eurl*> **fromStringNOCHECK** (QString *eurlstring*)

### Private Static Attributes

QHash<QChar, QString> **\_escapemap**  
 QMutex **\_escapemapmutex**  
 QMutex **\_mutex**  
 QHash<QString, std::weak\_ptr<const *Eurl*>> **\_eurluniverse**

## 10.1.152 Class sh::filesystem::EurlMisformattedException

**class** *sh::filesystem::EurlMisformattedException* : public *sh::exceptions::ArgumentException*  
 Shallot exception for invalid input in *Eurl* creation methods.

### Public Functions

**EurlMisformattedException** (*sh::exceptions::ExceptionData* *data*)  
 QString **message** () const  
 QString **name** () const  
 QString **classes** () const  
 QString **details** () const  
 QString **callstack** () const  
 QString **auxiliary** () const  
 QString **customValue** (QString *key*) const  
 bool **isRuntimeException** () const  
 bool **isProgramException** () const

```
bool isRetryable () const
bool isResumeable () const
bool isDetailsAreInteresting () const
int autoRetryRecommendedIn () const
void setResumeable (bool v)
void setReTryable (bool v)
void setCustomValue (QString key, QString value)
bool isClass (QString classname)
sh::exceptions::ExceptionData data ()
```

### Public Static Functions

```
template<class Handler>
std::shared_ptr<RegisterHandler<Handler>> createRegisterHandler (Handler handler, QS-
                                                                    tack<Handler> *stack)

void executeGuarded (std::function<void>)
    > fctint flags = 0 Executes some code with some standard exection handling around.
```

#### Parameters

- fct: The code to execute guarded.
- flags: Flags of *ExecuteGuardFlag* for choosing the behavior.

```
void executeGuarded_errorpanel (std::function<void>)
    > fctint flags = 0

QString _demangle (QString l)

void exceptionDialog (sh::exceptions::Exception &e, bool *doRetry)
```

### Public Static Attributes

```
const QString UNDEFINED_ERROR_OCCURRED = QObject::tr("An unspecified error occurred.")
const QString SHALLOT_MUST_CLOSE_TEXT = QObject::tr("Your Shallot process is disturbed by an error and needs to be c
const QString Value_isShallotException = "_isShallotException"
```

## 10.1.153 Class sh::filesystem::FilesystemHandler

```
class sh::filesystem::FilesystemHandler : public QObject
    Abstract base class for a custom filesystem handler.
```

Subclass and register it for implementing a new virtual filesystem, which lets new nodes appear somewhere in the filesystem tree and controls how to handle them.

Use *sh::filesystem::FilesystemHandlerRegister* for registration.

For executing some actions or checks on a filesystem, you should not use those handlers directly, but the higher-level *sh::filesystem::FilesystemOperation* class.



Subclassed by *sh::filesystemhandlers::ArchiveFilesystemHandler*, *sh::filesystemhandlers::GnomeIOFilesystemHandler*, *sh::filesystemhandlers::LocalFilesystemHandler*, *sh::filesystemhandlers::SharcFilesystemHandler*, *sh::scripting::api::ApiFilesystemHandler*, *sh::search::SearchFilesystemHandler*

## Public Functions

**FilesystemHandler** (*sh::filesystem::FilesystemModel \*model*)

Is (for subclasses) intended to be directly constructed and registered once.

**~FilesystemHandler** ()

void **itemlist** (*sh::filesystem::Operation \*op*, std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*, *sh::filesystem::FilesystemNodeType* *type*, *sh::filesystem::FilesystemNodeListEditor \*list*) = 0

Determine a list of subelements in a certain directory.

void **configureItems** (*sh::filesystem::Operation \*op*, QList<std::shared\_ptr<*FilesystemNode*>> *nodes*)

Configure newly created nodes (e.g. setting another icon or changing the display name).

*sh::filesystem::FilesystemNodeType* **getType** (*sh::filesystem::Operation \*op*, std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*) = 0

Determine if a file is regular, or a dir, or a link, ...

qint64 **getSize** (*sh::filesystem::Operation \*op*, std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*) = 0

Determine the size of a file.

QDateTime **getMtime** (*sh::filesystem::Operation \*op*, std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*) = 0

Determine the mtime of a file.

void **setMtime** (*sh::filesystem::Operation \*op*, std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*, QDateTime *time*) = 0

Set the mtime of a file.

QString **getMimetype** (*sh::filesystem::Operation \*op*, std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*) = 0

Determine mime type of a file.

QString **getLinkTarget** (*sh::filesystem::Operation \*op*, std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*) = 0

Resolve a link.

QList<QString> **listExtendedAttributes** (*sh::filesystem::Operation \*op*, std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*)

Fetches the list of available extended attributes for an entry.

quint64 **getExtendedAttributeSize** (*sh::filesystem::Operation \*op*, std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*, QString *attribute*)

Returns the size of value for one extended attribute for an entry.

QByteArray **getExtendedAttribute** (*sh::filesystem::Operation \*op*, std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*, QString *attribute*)

Returns the value for one extended attribute for an entry.

void **setExtendedAttribute** (*sh::filesystem::Operation \*op*, std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*, QString *attribute*, QByteArray *value*)

Sets the value for one extended attribute for an entry.

void **removeExtendedAttribute** (*sh::filesystem::Operation \*op*, std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*, QString *attribute*)

Removes one extended attributes for an entry.

```
QMap<QString, QString> getCustomAttributes (sh::filesystem::Operation *op,
                                              std::shared_ptr<const sh::filesystem::Eurl>
                                              eurl)
```

Returns the custom attributes for an entry.

In contrast to extended attributes, the custom attributes are not directly stored in the filesystem this way, but handle implementation specific aspects (like permissions).

```
void setCustomAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
                        sh::filesystem::Eurl> eurl, QString attribute, QString value)
```

Sets the value for one custom attribute for an entry.

See also getCustomAttributes.

```
bool canGetFileContent (sh::filesystem::Operation *op, std::shared_ptr<const
                        sh::filesystem::Eurl> eurl) = 0
```

Returns if it is allowed to get the content of a certain file.

```
std::shared_ptr<QIODevice> getFileContent (sh::filesystem::Operation *op,
                                           std::shared_ptr<const sh::filesystem::Eurl> eurl) =
0
```

Get the content of a file.

```
bool canCreateDirectory (sh::filesystem::Operation *op, std::shared_ptr<const
                        sh::filesystem::Eurl> eurl) = 0
```

Returns if it is allowed to create subdirectories in a certain directory.

```
void createDirectory (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                      eurl) = 0
```

Create a directory.

```
bool canCreateLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl) = 0
```

Returns if it is allowed to create a link in a certain directory.

```
void createLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
                 QString target) = 0
```

Create a link.

```
bool canCreateFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl) = 0
```

Returns if it is allowed to create files in a certain directory.

```
void createFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
                 QIODevice *content, HandlerTransfer *handlertransfer) = 0
```

Creates a file with some content.

It may use handlertransfer for some better integration, like cancel support and progress monitoring.

```
bool canDeleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl) = 0
```

Returns if it is allowed to delete a certain file/directory/link/...

```
void deleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl) =
0
```

Delete a file/directory/link/...

```
void deleteDirectoryIfEmpty (sh::filesystem::Operation *op, std::shared_ptr<const
                             sh::filesystem::Eurl> eurl)
```

Deletes a directory only if it is empty.

```
bool canRenameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    src) = 0
```

Returns if it is allowed to rename a certain file, directory, link, ... (see renameItem).

void **renameItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> src, QString destpath) = 0  
 Renames an item to another path.

It can be a simple filename change or also changes in the deeper path segments.

QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> **getActions** (const QList<std::shared\_ptr<const *sh::filesystem::Eurl*>> eurls) = 0  
 Returns a list of actions (see former example) for showing for certain files.

bool **requiresResolvedLinks** ()  
 Returns if this handler requires to be used by the engine with resolved links.

void **viewEnteredDirectory** (std::shared\_ptr<const *sh::filesystem::Eurl*>)  
 Callback for preparing stuff when the view entered some directory (e.g. for enabling watchers). See also viewLeftDirectory.

void **viewLeftDirectory** (std::shared\_ptr<const *sh::filesystem::Eurl*>)  
 Callback for preparing stuff when the view left some directory. See also viewEnteredDirectory.

bool **isWellKnownScheme** ()  
 Returns if the scheme for this handler is a well known one (which external programs typically would understand).

void **search** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, std::function<void> std::shared\_ptr<const *sh::filesystem::Eurl*>, QList<std::shared\_ptr<*sh::search::SearchCriterion*>>, *sh::filesystem::FilesystemNodeType* ftype > addfile, std::function<void>std::shared\_ptr<const *sh::filesystem::Eurl*>> visitdir, QList<std::shared\_ptr<*sh::search::SearchCriterion*>> criteria) Helps searching for files.

Override this method with a behavior identical to the default, if a specialized implementation can be much faster than the default one.

void **customizeUi** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> node, bool \*showSearchPanel)  
 Creates a custom gui, which becomes part of the fileview.

*sh::filesystem::FilesystemModel* \***model** ()  
 A convenience shortcut to the *sh::filesystem::FilesystemModel* (a singleton).

**class HandlerTransfer**  
 Subclassed by *sh::filesystem::FilesystemOperation::MyHandlerTransfer*,  
*sh::filesystem::FilesystemOperationTransfers::SingleStepMonitor*

## Public Functions

void **respectCancel** ()

void **incrementTransferredBytes** (qint64 donebytes)

**~HandlerTransfer** ()

### 10.1.154 Class `sh::filesystem::FileSystemHandlerRegister`

**class** `sh::filesystem::FileSystemHandlerRegister` : **public** `QObject`, **public** `sh::base::Singleton`  
A register of filesystem handlers.

Each active (i.e. referred to by existing nodes) instance of `sh::filesystem::FileSystemHandler` must be registered here.

#### Public Functions

**void** `addHandler` (`QString` *scheme*, `std::shared_ptr<sh::filesystem::FileSystemHandler>` *handler*)  
Registers a filesystem handler.

#### Parameters

- *scheme*: The scheme (very first part of a `sh::filesystem::Eurl`) for which the handler is responsible for.
- *handler*: The filesystem handler.

`std::shared_ptr<sh::filesystem::FileSystemHandler>` `findHandler` (`QString` *scheme*)  
Finds a filesystem handler by scheme.

**void** `doInitialize` ()  
Executes singleton initialization.

**void** `doShutdown` ()  
Executes singleton shutdown.

**void** `shutdown` ()  
Shutdown down this singleton.

**bool** `isAlive` ()  
Returns if this singleton is alive (true until its shutdown begins).

#### Private Functions

`FileSystemHandlerRegister` ()

#### Private Members

`QHash<QString, std::shared_ptr<sh::filesystem::FileSystemHandler>>` `handlers`

`QMutex` `mutex`

### 10.1.155 Class `sh::filesystem::FileSystemHandler::HandlerTransfer`

**class** `sh::filesystem::FileSystemHandler::HandlerTransfer`  
Subclassed by `sh::filesystem::FileSystemOperation::MyHandlerTransfer`, `sh::filesystem::FileSystemOperationTransfers::SingleStateTransfer`

## Public Functions

```
void respectCancel ()
void incrementTransferredBytes (qint64 donebytes)
~HandlerTransfer ()
```

### 10.1.156 Class `sh::filesystem::FilesystemModel`

**class** `sh::filesystem::FilesystemModel` : **public** `QAbstractItemModel`, **public** `sh::base::Singleton`  
The filesystem model.

This is the engine which creates and manages `sh::filesystem::FilesystemNode` nodes. Filesystem nodes are used on many places for all kinds of operations.

## Public Functions

```
std::shared_ptr<sh::filesystem::FilesystemNode> rootNode ()
    The sh::filesystem::FilesystemNode which is the root node of the entire model. It is the parent for all
    toplevel nodes.
```

```
QVariant data (const QModelIndex &index, int role = Qt::DisplayRole) const
Qt::ItemFlags flags (const QModelIndex &index) const
QVariant headerData (int section, Qt::Orientation orientation, int role = Qt::DisplayRole) const
QModelIndex index (int row, int column, const QModelIndex &parent = QModelIndex()) const
QModelIndex parent (const QModelIndex &index) const
int rowCount (const QModelIndex &parent = QModelIndex()) const
int columnCount (const QModelIndex &parent = QModelIndex()) const
sh::filesystem::FilesystemModelFileviewProxy *createFileviewProxy (QModelIndex root,
                                                                    bool withtooltip,
                                                                    std::function<void> std::shared_ptr<sh::filesystem::FilesystemModelFileviewProxy> &onBecomesInvalid) const
    > onBecomesInvalid Creates a sh::filesystem::FilesystemModelFileviewProxy for presenting the content of
    a directory.
```

```
QList<QPersistentModelIndex> findIndexesForEurl (std::shared_ptr<const
                                                sh::filesystem::Eurl> eurl)
    Get a list of qt model indexes for a sh::filesystem::Eurl.
    If nodes for this eurl are unknown to the model so far, it tries to build them. In typical cases, this list either
    contains one element, or is empty if the filesystem handlers decide that this file does not exist. But in some
    cases, there is also more than one index for one sh::filesystem::Eurl (when the sh::filesystem::Eurl appears
    on more than one place in the tree).
```

```
QList<std::shared_ptr<sh::filesystem::FilesystemNode>> findNodesForEurl (std::shared_ptr<const
                                                                    sh::filesystem::Eurl>
                                                                    eurl)
    Get a list of sh::filesystem::FilesystemNode for a sh::filesystem::Eurl.
    If it is unknown to the model so far, it tries to build them. In typical cases, this list either contains one
    element, or is empty if the filesystem handlers decide that this file does not exist. But in some cases, there
    is also more than one node for one sh::filesystem::Eurl (when the sh::filesystem::Eurl appears on more
```

than one place in the tree). It only returns nodes, which are ‘alive’, i.e. which have a living parent and which are a child of this parent.

`QList<std::shared_ptr<sh::filesystem::FileSystemNode>> tryGetNodesForEurl (std::shared_ptr<const sh::filesystem::Eurl> eurl)`

Returns a list of `sh::filesystem::FileSystemNode` for a `sh::filesystem::Eurl`. It only considers the current state of the in-memory model. It will only return nodes which are already known to the model so far. This operation is cheaper and handling only the known nodes is sufficient in many situations. In typical cases, this list either contains one element, or is empty. But in some cases, there is also more than one node for one `sh::filesystem::Eurl` (when the `sh::filesystem::Eurl` appears on more than one place in the tree). It only returns nodes, which are ‘alive’, i.e. which have a living parent and which are a child of this parent.

`std::shared_ptr<sh::filesystem::FileSystemNode> getNodeForIndex (const QModelIndex index) const`

Returns the `sh::filesystem::FileSystemNode` for a qt model index (in the main model).

`QModelIndex getIndexForNode (std::shared_ptr<sh::filesystem::FileSystemNode> node)`

Returns a qt model index (in the main model) for a `sh::filesystem::FileSystemNode`.

`std::shared_ptr<sh::filesystem::FileSystemNode> createFileSystemNode (std::shared_ptr<const sh::filesystem::Eurl> eurl, sh::filesystem::FileSystemHandler *handler, sh::filesystem::FileSystemNodeType nodetype, bool isHidden, std::shared_ptr<sh::filesystem::FileSystemNode> parent, bool doinsert = true, bool showInitialLoadingLabel = true)`

Creates a new `sh::filesystem::FileSystemNode` for placing it into the model.

If such a node (with the same eurl for the same parent node) does not exist, it generates a new one. If there already is such a node alive, but currently not placed in the model, it recycles this one. This can happen when references exist to a node which is not yet inserted or which is removed meanwhile. It is not allowed to call this method when such a node already exists in the model.

Use this function for getting a `sh::filesystem::FileSystemNode`, which is to be added to the model now or later. Depending on some parameter values, a call directly adds the node to the filesystem model (not e.g. when `doinsert` is `false` or `parentnode` is 0) It is typically used within a `sh::filesystem::FileSystemHandler` implementation.

`std::shared_ptr<sh::filesystem::FileSystemNode> getOrCreateFileSystemNode (std::shared_ptr<const sh::filesystem::Eurl> neweurl, sh::filesystem::FileSystemHandler *handler, sh::filesystem::FileSystemNodeType nodetype, bool isHidden, std::shared_ptr<sh::filesystem::FileSystemNode> parent, bool doinsert = true, bool showInitialLoadingLabel = true, bool *pIsNew = 0)`

Returns a *sh::filesystem::FilesystemNode* for using it as a child node in the model.

It either creates a new one, if there currently is no node for this eurl in this parentnode, or returns the existing one. Even for existing ones, this call can change the nodetype of that node.

Depending on some parameter values, a call directly adds the node to the filesystem model (not e.g. when `doinsert` is `false` or `parent` is 0).

```
void refreshData (std::shared_ptr<const sh::filesystem::Eurl> eurl, bool forceFindParent = false,
                 bool withDetails = true)
```

Request to refresh the internal model information for a *sh::filesystem::Eurl*. This may be a place which is already known (then a change of some metadata or the removal is detected) or a formerly unknown place (then new nodes get inserted in the model).

```
void addOpenNodeHelper (int index, std::function<QList<std::shared_ptr<sh::filesystem::FilesystemNode>>> std::shared_ptr<
                        sh::filesystem::Eurl>
```

> *openNodeHelper* Registers a helper method for ‘opening’ (mounting, activating, ...) locations on demand.

Only used in very rare cases.

```
~FilesystemModel ()
```

```
void doInitialize ()
```

Executes singleton initialization.

```
void doShutdown ()
```

Executes singleton shutdown.

```
void shutdown ()
```

Shutdown down this singleton.

```
bool isAlive ()
```

Returns if this singleton is alive (true until its shutdown begins).

## Private Functions

```
FilesystemModel ()
```

```
QList<std::shared_ptr<sh::filesystem::FilesystemNode>> _findNodesForEurl_helper (std::shared_ptr<const
                                                                              sh::filesystem::Eurl>
                                                                              eurl)
```

```
QList<std::shared_ptr<sh::filesystem::FilesystemNode>> openRootEurl (std::shared_ptr<const
                                                                      sh::filesystem::Eurl> eurl)
```

```
void _subitemFetchingStateChanged (std::shared_ptr<sh::filesystem::FilesystemNode> node,
                                   bool value)
```

## Private Members

```
QList< std::function< QList< std::shared_ptr< sh::filesystem::FilesystemNode > >std::s
```

```
QMap< int, std::function< QList< std::shared_ptr< sh::filesystem::FilesystemNode > >std
```

```
std::shared_ptr<sh::filesystem::FilesystemNode> rootnode
```

```
QMutex mutex
```

```
QHash<std::shared_ptr<const sh::filesystem::Eurl>, std::weak_ptr<sh::filesystem::FilesystemNode>> eurl2node
```

```
QMutex eurl2nodemutex
```

QMutex **\_nodeDataMutex**

QMutex **\_openNodeHelpersMutex**

### Friends

**friend class** FilesystemNode

**friend class** LoadOnDemandPlaceholderFilesystemNode

**friend class** ModelBackedFilesystemNodeList

**friend class** ::sh::filesystem::DetailColumn

## 10.1.157 Class sh::filesystem::FilesystemModelDirectoryTreeProxy

**class** *sh::filesystem::FilesystemModelDirectoryTreeProxy* : public QSortFilterProxyModel, public *sh::ba*

A filesystem proxy model for the directory tree.

It has a special sorting and filtering behavior.

Used internally, mostly for the user interface.

### Public Functions

void **enforceVisibility** (std::shared\_ptr<*FilesystemNode*> *node*)

Marks a node as visible in the directory tree, even if it is a hidden node.

void **deEnforceVisibility** (std::shared\_ptr<*FilesystemNode*> *node*)

Unmarks a node for being visible even if hidden (reverses *enforceVisibility()*).

void **doShutdown** ()

Executes singleton shutdown.

void **shutdown** ()

Shutdown down this singleton.

bool **isAlive** ()

Returns if this singleton is alive (true until its shutdown begins).

### Private Functions

**FilesystemModelDirectoryTreeProxy** ()

## 10.1.158 Class sh::filesystem::FilesystemModelDirectoryTreeProxyVisibilityEnforcements

**class** *sh::filesystem::FilesystemModelDirectoryTreeProxyVisibilityEnforcements* : public QObject

Maintains a list of currently visible directories (i.e. the current one in each view) and controls enforced visibility of them in the directory tree.



## Public Functions

void **nodeEnteredView** (std::shared\_ptr<*FilesystemNode*> *node*)  
 Called when a view enters the given directory node.

void **nodeLeftView** (std::shared\_ptr<*FilesystemNode*> *node*)  
 Called when a view leaves the given directory node.

void **nodeCollapsedInTree** (std::shared\_ptr<*FilesystemNode*> *node*)  
 Called when a directory node is collapsed in the directory tree.

void **doShutdown** ()  
 Executes singleton shutdown.

void **shutdown** ()  
 Shutdown down this singleton.

bool **isAlive** ()  
 Returns if this singleton is alive (true until its shutdown begins).

## Private Functions

**FilesystemModelDirectoryTreeProxyVisibilityEnforcements** ()

bool **\_addenforcement** (std::shared\_ptr<*FilesystemNode*> *node*, std::shared\_ptr<*FilesystemNode*> *hnode*)

## Private Members

QMultiMap<std::shared\_ptr<*FilesystemNode*>, QObject\*> **\_enterednodesrepresentatives**

QList<std::tuple<std::weak\_ptr<*FilesystemNode*>, QList<std::shared\_ptr<*FilesystemNode*>>>> **hiddenForcedVisibles**

### 10.1.159 Class sh::filesystem::FilesystemModelFileviewProxy

**class** *sh::filesystem::FilesystemModelFileviewProxy* : **public** QSortFilterProxyModel

A filesystem proxy model for a fileview.

It regards the sorting and filtering behavior set up for the connected fileview.

Used internally, mostly for the user interface.

## Public Functions

**FilesystemModelFileviewProxy** (QModelIndex *root*, bool *withtooltip*, QObject \**parent* = 0)  
 Constructed only indirectly.

void **setSizeFormattingMode** (*SizeFormatting mode*)

*SizeFormatting* **getSizeFormattingMode** () **const**

void **setHiddenFilesVisible** (bool *v*)

bool **getHiddenFilesVisible** () **const**

QVariant **data** (**const** QModelIndex &*index*, int *role* = Qt::DisplayRole) **const**

void **triggerReloadData** ()

```
void setThumbnail (bool enabled, int size = 32)
bool getThumbnailEnabled ()
int getThumbnailSize ()
```

### 10.1.160 Class `sh::filesystem::FilesystemModelSubtreeProxy`

**class** `sh::filesystem::FilesystemModelSubtreeProxy` : **public** QAbstractItemModel

A filesystem proxy model for setting a new root node to an existing fileview proxy.

Used internally, mostly for the user interface.

#### Public Functions

**FilesystemModelSubtreeProxy** (QModelIndex *root*, `sh::filesystem::FilesystemModelFileviewProxy` \**upperproxy*)

Constructed only indirectly.

QVariant **data** (const QModelIndex &*index*, int *role*) **const**

Qt::ItemFlags **flags** (const QModelIndex &*index*) **const**

QVariant **headerData** (int *section*, Qt::Orientation *orientation*, int *role* = Qt::DisplayRole) **const**

QModelIndex **index** (int *row*, int *column*, const QModelIndex &*parent* = QModelIndex()) **const**

QModelIndex **parent** (const QModelIndex &*index*) **const**

int **rowCount** (const QModelIndex &*parent* = QModelIndex()) **const**

int **columnCount** (const QModelIndex &*parent* = QModelIndex()) **const**

QModelIndex **mapFromSource** (const QModelIndex *mi*) **const**

QModelIndex **mapToSource** (const QModelIndex *mi*) **const**

#### Signals

void **becomesInvalid** (std::shared\_ptr<`sh::filesystem::FilesystemNode`> *bestValid*)

#### Private Members

bool **\_inChangeTransaction**

std::shared\_ptr<`sh::filesystem::FilesystemNode`> **\_rootnode**

QPersistentModelIndex **\_rootindex**

`sh::filesystem::FilesystemModel` \***mainmodel**

`sh::filesystem::FilesystemModelFileviewProxy` \***\_upperproxy**

### 10.1.161 Class `sh::filesystem::FileSystemNode`

**class** `sh::filesystem::FileSystemNode` : **public** QObject, **public** std::enable\_shared\_from\_this<*FileSystemNode*>

A representation of a location in the filesystem tree.

It represents filesystem nodes like a file or a directory in *sh::filesystem::FileSystemModel*.

Please note: Instances can be associated with any kind of elements in the filesystem (files, directories, links, ...). It might also point to something which does not exist at all (see `parentnode`). The documentation sometimes explicitly makes a difference between those kinds (files, directories, links, ...; often called ‘node type’). But it often uses the term ‘file’ implicitly while meaning all kinds of elements; assuming that e.g. a directory is just a special kind of a file. It should be clear from the particular context which meaning applies.

Subclassed by *sh::filesystem::LoadOnDemandPlaceholderFileSystemNode*

#### Public Functions

**~FileSystemNode** ()

std::shared\_ptr<const *sh::filesystem::Eurl*> **eurl** ()

Returns the *sh::filesystem::Eurl* entry address.

*sh::filesystem::FileSystemModel* \***model** ()

Convenience shortcut to the *sh::filesystem::FileSystemModel*.

QString **displayName** ()

Returns the display name (what the user interface displays).

void **setDisplayName** (QString *displayname*)

Sets the display name (what the user interface displays).

*sh::filesystem::FileSystemHandler* \***handler** ()

Returns the handler which is responsible for this node. This should be the same as *sh::filesystem::FileSystemHandlerRegister.findHandler*(eurl.scheme).

int **nodetype** ()

Returns the `FileSystemNodeType` node type.

int **childnodeCount** ()

Returns the number of children nodes.

std::shared\_ptr<*sh::filesystem::FileSystemNode*> **childnode** (int *i*)

Returns *i*-th child node.

*sh::filesystem::ModelBackedFileSystemNodeList* \***childnodes** ()

Returns the list of children.

std::shared\_ptr<*sh::filesystem::FileSystemNode*> **parentnode** () const

Returns the parent node.

This is 0 if the node does not exist in the model (i.e. not yet inserted or removed afterwards). In most cases, this can be interpreted as ‘file currently does not exist’.

int **index** ()

Returns the index of this node in the parent’s list of children.

void **addChild** (std::shared\_ptr<*sh::filesystem::FileSystemNode*> *node*)

Adds a new child. Not allowed to call more than once for a node.

void **removeChild** (std::shared\_ptr<*sh::filesystem::FileSystemNode*> *node*)

Removes a new child. Not allowed to call more than once for a node.

bool **isFetchingSubitems** ()

Checks if currently subitems are fetched (i.e. the ‘loading’ node is shown).

QIcon **icon** ()

Returns the node icon.

void **setIcon** (QIcon *icon*)

Sets the node icon.

bool **isHidden** ()

Returns if the node is hidden.

Note: It’s not difficult for the user to also show the hidden items.

void **setHidden** (bool *v*)

Sets if the node is hidden.

See also *isHidden()*.

void **addDetail** (std::shared\_ptr<*sh::filesystem::DetailColumn*> *column*)

Adds a detail to this node.

std::shared\_ptr<*sh::filesystem::DetailColumn*> **getDetailColumnByIndex** (int *index*)

Returns the i-th detail column registered to this node.

int **getDetailColumnsCount** () const

Returns the number of detail columns registered to this node.

bool **isRootNode** () const

Checks if this is the root node of the model.

bool **isConfigured** () const

Checks if this node was already configured by a handler.

void **requestDetails** (bool *force* = true)

Requests to fetch details for this node.

bool **isAlive** () const

Checks if this node currently exists in the model.

QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> **linkDestinationNodes** () const

Returns the list of link source nodes. If this node is a link, the link destination nodes may influence the behavior and appearance of this node.

This information does not come from inside but must be set from outside the model implementation.

void **setLinkDestinationNodes** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>>  
*linkdestinations*)

Sets the list of link destination nodes. If this node is a link, the link destination nodes may influence the behavior and appearance of this node.

void **requestContainedItems** (*sh::filesystem::FilesystemNodeType* *type* = *FilesystemNode-  
Type::NONE*)

Requests to fetch the children of this node with help of the filesystem handler.

**FilesystemNode** (*sh::filesystem::FilesystemModel* \**model*, std::shared\_ptr<const  
*sh::filesystem::Eurl*> *eurl*, QString *displayname*, *sh::filesystem::FilesystemHandler*  
\**handler*, *FilesystemNodeType* *nodetype*, bool *ishidden*)

Constructed only by the infrastructure and made available otherwise.

## Signals

void **removed** ()  
Is emitted when this node is not placed in the tree anymore. This does not mean the physical deletion of the instance, but just means it is not alive from now on.

void **\_detailsAvailable** ()  
Is emitted when values for detail columns arrived.

void **iconChanged** ()  
Is emitted when the icon changed.

void **isHiddenChanged** ()  
Is emitted when the hidden flag changed.

## Private Functions

void **setDetail** (std::shared\_ptr<sh::filesystem::DetailColumn> column, QVariant value)

int **getInsertPositionForDetailColumn** (std::shared\_ptr<sh::filesystem::DetailColumn> newCol)

## Friends

**friend class** sh::filesystem::FilesystemModel

**friend class** sh::filesystem::DetailColumn

**friend class** LoadOnDemandPlaceholderFilesystemNode

**friend class** sh::filesystem::FilesystemModelFileviewProxy

**friend class** sh::filesystem::FilesystemModelSubtreeProxy

**friend class** sh::filesystem::ModelBackedFilesystemNodeList

### 10.1.162 Class sh::filesystem::FilesystemNodeList

**class** sh::filesystem::FilesystemNodeList : public QObject

A data structure for children lists of filesystem nodes.

Different subclasses exist with different behaviors and use cases.

Subclassed by sh::filesystem::AdhocFilesystemNodeList, sh::filesystem::ModelBackedFilesystemNodeList

## Public Functions

void **addItems** (QSet<std::shared\_ptr<sh::filesystem::FilesystemNode>> nodes) = 0  
Adds items to this list. In a model-backed list, this triggers the internal model mounting calls. It is not allowed to call this method twice with the same node.

void **addItem** (std::shared\_ptr<sh::filesystem::FilesystemNode> node) = 0  
Adds an item to this list. In a model-backed list, this triggers the internal model mounting calls. It is not allowed to call this method twice with the same node.

void **removeItems** (QSet<std::shared\_ptr<sh::filesystem::FilesystemNode>> nodes) = 0  
Removes nodes from this list. In a model-backed list, this triggers the internal model unmounting calls. It is not allowed to call this method with a node, which is not contained in that list.

void **removeItem** (std::shared\_ptr<sh::filesystem::FilesystemNode> node) = 0  
Removes a node from this list. In a model-backed list, this triggers the internal model unmounting calls.  
It is not allowed to call this method with a node, which is not contained in that list.

void **resetItems** (sh::filesystem::FilesystemNodeType type, QSet<std::shared\_ptr<sh::filesystem::FilesystemNode>>  
nodes) = 0  
Resets the list for a given node type to a given new list of children nodes.

std::shared\_ptr<sh::filesystem::FilesystemNode> **myNode** () = 0  
Returns the node which owns this children list. The result may be 0 for non model-backed lists.

bool **contains** (std::shared\_ptr<FilesystemNode> node)  
Checks if this list contains a given node.

const QList<std::shared\_ptr<sh::filesystem::FilesystemNode>> **\*nodes** ()  
Returns the list of nodes currently stored in this instance.

### 10.1.163 Class sh::filesystem::FilesystemNodeListEditor

**class sh::filesystem::FilesystemNodeListEditor**

A list editor for easily modifying a *FilesystemNodeList*.

Mainly used for listing child nodes in *sh::filesystem::FilesystemHandler.itemlist*. In most cases, you should just use *setItems*.

#### Public Functions

**FilesystemNodeListEditor** (*FilesystemNodeList* \*list, std::shared\_ptr<const  
sh::filesystem::Eurl> parenteurl, sh::filesystem::FilesystemNodeType  
nodetype)

Constructed only by the infrastructure and made available otherwise.

**~FilesystemNodeListEditor** ()

void **setItems** (QList<QString> items)

Sets a new list content. This function is all you need in most situations. For iteratively adding nodes, which makes the intermediate results visible in the user interface, see *addItem*.

void **beginIterativeAdding** ()

Must be called before you begin to iteratively fill the children list (e.g. with *addItem*). You must also call *endIterativeAdding* afterwards.

void **addItem** (QString item)

Adds a new children to the list, directly showing that in the user interface, while you can proceed filling the list. Please read *beginIterativeAdding* as well! In most cases, you don't need this function.

std::shared\_ptr<FilesystemNode> **addCustomItem** (std::shared\_ptr<const sh::filesystem::Eurl>  
eurl, QString displayname = QString(),  
sh::filesystem::FilesystemHandler \*handler =  
0)

Adds a new children to the list, directly showing that in the user interface, while you can proceed filling the list. Please read *beginIterativeAdding* as well! In most cases, you don't need this function.

Note: If you specify a handler, it must be the one matching to *eurl*'s scheme.

void **addExistingNodeItem** (std::shared\_ptr<sh::filesystem::FilesystemNode> node)

Adds a new children to the list, directly showing that in the user interface, while you can proceed filling the list. Please read *beginIterativeAdding* as well! In most cases, you don't need this function.

void **endIterativeAdding** ()

Must be called after you iteratively filled the children list with addItem. This automatically removes all the old nodes, which you haven't added in this session.

void **setItemsAreHosts** ()

After this call, the addItem method will consider the given names as hostnames instead of new path segments. The resulting *sh::filesystem::Eurl* will differ accordingly. This only makes sense as children for root eurls.

*FilesystemNodeList* \***rawlist** ()

Returns the *FilesystemNodeList* backend. You should rarely need it.

## Private Functions

std::shared\_ptr<const *sh::filesystem::Eurl*> **getChildEurl** (std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QString item)

## Private Members

*FilesystemNodeList* \***\_list**

std::shared\_ptr<const *sh::filesystem::Eurl*> **\_parenteurl**

*sh::filesystem::FilesystemNodeType* **\_nodetype**

std::shared\_ptr<*sh::filesystem::FilesystemNode*> **\_parentnode**

*sh::filesystem::FilesystemModel* \***\_model**

*sh::filesystem::FilesystemHandler* \***\_handler**

bool **\_itemsAreHosts** = false

bool **\_beganAddingIteratively** = false

QSet<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> **\_iterativeAddingsBeforeState**

## Private Static Attributes

QSet<*FilesystemNodeList*\*> **\_pendingIterativeAddings**

QMutex **\_pendingMutex**

QWaitCondition **\_iterativeAddingPossibleCondition**

### 10.1.164 Class *sh::filesystem::FilesystemNode::DetailsRequestor*

**class** *sh::filesystem::FilesystemNode::DetailsRequestor*

Internal engine for multithreaded requesting of fetching all details for a node.

## Public Functions

**DetailsRequestor** ()

void **requestDetails** (std::shared\_ptr<sh::filesystem::FilesystemNode> node)  
Fetch details for a node asynchronously in background.

## Public Members

QQueue<std::shared\_ptr<sh::filesystem::FilesystemNode>> **nodes**

QMutex **nodesmutex**

## Private Members

int **availthreads**

### 10.1.165 Class sh::filesystem::FilesystemOperation

**class** sh::filesystem::FilesystemOperation

A high-level interface for filesystem operations.

It is always based on the transaction of a *Operation* instance (which also gives you access to a *FilesystemOperation* object).

Some calls optionally allow to give an own instance of *sh::filesystem::FilesystemOperationProgressMonitor* for some additional functionality. Please note that not all calls provide all those functionality (some do not use e.g. the conflict resolution at all). If not provided, a default behavior is applied.

## Public Functions

**FilesystemOperation** (sh::filesystem::Operation \*operation)

Constructed only by the infrastructure and made available otherwise.

QList<std::shared\_ptr<const sh::filesystem::Eurl>> **itemlist** (std::shared\_ptr<const  
sh::filesystem::Eurl> eurl,  
sh::filesystem::FilesystemNodeType  
type =  
sh::filesystem::FilesystemNodeType::NONE)

Gets a list of shallot.sh::filesystem::FilesystemNode nodes in a directory (optionally filter by given types).

sh::filesystem::FilesystemNodeType **getType** (std::shared\_ptr<const sh::filesystem::Eurl> eurl)

Gets the shallot.sh::filesystem::FilesystemNodeType node type for an entry.

qint64 **getFileSize** (std::shared\_ptr<const sh::filesystem::Eurl> eurl)

Gets the file size for an entry.

QString **getLinkTarget** (std::shared\_ptr<const sh::filesystem::Eurl> eurl)

Gets the link target for an entry (if it is a link).

bool **canGetFileContent** (std::shared\_ptr<const sh::filesystem::Eurl> eurl)

Can we get the file content for an entry?

std::shared\_ptr<QIODevice> **getFileContent** (std::shared\_ptr<const sh::filesystem::Eurl> eurl)

Gets the file content for an entry as QIODevice.



bool **canCreateDirectory** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*)  
Can we create a given directory?

void **createDirectory** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*,  
std::shared\_ptr<*sh::filesystem::FilesystemOperationProgressMonitor*>  
*progressmon* = 0)  
Creates a directory.

#### Parameters

- *progressmon*: An optional progress monitor.

bool **canCreateLink** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*)  
Can we create a given link?

void **createLink** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*, QString *target*,  
std::shared\_ptr<*sh::filesystem::FilesystemOperationProgressMonitor*> *progressmon*  
= 0)  
Creates a link.

#### Parameters

- *progressmon*: An optional progress monitor.

bool **canCreateFile** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*)  
Can we create a given file?

void **createFile** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*, QIODevice *\*content*,  
std::shared\_ptr<*sh::filesystem::FilesystemOperationProgressMonitor*> *progressmon*  
= 0)  
Creates a file.

#### Parameters

- *progressmon*: An optional progress monitor.

bool **canDeleteItem** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*)  
Can we delete a given entry?

void **deleteItem** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*,  
std::shared\_ptr<*sh::filesystem::FilesystemOperationProgressMonitor*> *progressmon*  
= 0)  
Delete an entry.

#### Parameters

- *progressmon*: An optional progress monitor.

void **deleteDirectoryIfEmpty** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*,  
std::shared\_ptr<*sh::filesystem::FilesystemOperationProgressMonitor*>  
*progressmon* = 0)  
Delete a directory entry if empty.

#### Parameters

- *progressmon*: An optional progress monitor.

```
bool canMoveItem (std::shared_ptr<const sh::filesystem::Eurl> src, std::shared_ptr<const  
                  sh::filesystem::Eurl> dest = 0)
```

Checks if it is allowed to move a certain item.

If the destination is known as well, it might help to pass it as well.

This does not guarantee success in the transfer, but is just a cheap early check.

```
void moveItems (QList<std::shared_ptr<const sh::filesystem::Eurl>>  
                src, std::shared_ptr<const sh::filesystem::Eurl> dest,  
                std::shared_ptr<sh::filesystem::FilesystemOperationProgressMonitor> progress-  
                mon = 0)
```

Moves entries.

#### Parameters

- dest: The requested new common parent directory.
- progressmon: An optional progress monitor.

```
void moveItem (std::shared_ptr<const sh::filesystem::Eurl> src, std::shared_ptr<const  
                sh::filesystem::Eurl> dest, std::shared_ptr<sh::filesystem::FilesystemOperationProgressMonitor>  
                progressmon = 0)
```

Moves an entry.

#### Parameters

- dest: The new location (not the new parent).
- progressmon: An optional progress monitor.

```
bool canCopyItem (std::shared_ptr<const sh::filesystem::Eurl> src, std::shared_ptr<const Eurl>  
                  dest = 0)
```

Checks if it is allowed to copy a certain item.

If the destination is known as well, it might help to pass it as well.

This does not guarantee success in the transfer, but is just a cheap early check.

```
void copyItems (QList<std::shared_ptr<const sh::filesystem::Eurl>>  
                src, std::shared_ptr<const sh::filesystem::Eurl> dest,  
                std::shared_ptr<sh::filesystem::FilesystemOperationProgressMonitor> progress-  
                mon = 0)
```

Copies entries.

#### Parameters

- dest: The requested new common parent directory.
- progressmon: An optional progress monitor.

```
void copyItem (std::shared_ptr<const sh::filesystem::Eurl> src, std::shared_ptr<const  
                sh::filesystem::Eurl> dest, std::shared_ptr<sh::filesystem::FilesystemOperationProgressMonitor>  
                progressmon = 0)
```

Copies an entry.

#### Parameters

- dest: The new location (not the new parent).
- progressmon: An optional progress monitor.

QList<std::shared\_ptr<*FileSystemNode*>> **resolveNodeLink** (std::shared\_ptr<*FileSystemNode*>  
*node*, bool *recursive* = true, bool  
*excludeOwn* = false)

Resolve a link as node with or without recursion.

std::shared\_ptr<const *sh::filesystem::Eurl*> **resolveEurlLink** (std::shared\_ptr<const  
*sh::filesystem::Eurl*> *eurl*, bool  
*recursive* = true, bool *excludeOwn*  
= false, int *tries* = 100)

Resolve a link as *sh::filesystem::Eurl* with or without recursion.

QList<QString> **listExtendedAttributes** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*)

quint64 **getExtendedAttributeSize** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*, QString  
*attribute*)

QByteArray **getExtendedAttribute** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*, QString  
*attribute*)

void **setExtendedAttribute** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*, QString *attribute*,  
QByteArray *value*)

void **removeExtendedAttribute** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*, QString *at-*  
*tribute*)

## Public Static Functions

void **addTransferrableDetailColumn** (int *index*, std::shared\_ptr<*sh::filesystem::DetailColumn*>  
*detailColumn*)

Registers a detail column for transferring it when file transfer take place.

### Parameters

- *index*: An integer which controls the order of transferring.

QList<std::shared\_ptr<*sh::filesystem::DetailColumn*>> **transferrableDetailColumns** ()

A list of all detail columns which are registered becoming transferred in file transfers.

## Private Functions

std::shared\_ptr<*sh::filesystem::FileSystemHandler*> **\_handler** (std::shared\_ptr<const  
*sh::filesystem::Eurl*> *eurl*)

std::shared\_ptr<*sh::filesystem::FileSystemOperationProgressMonitor*> **\_monitor** (std::shared\_ptr<*sh::filesystem::FileSystemOp-*  
*m*)

std::shared\_ptr<const *sh::filesystem::Eurl*> **\_resolveIfNeeded** (std::shared\_ptr<*sh::filesystem::FileSystemHandler*>  
*handler*, std::shared\_ptr<const  
*sh::filesystem::Eurl*> *eurl*)

### Private Members

*Operation* \*\_operation

### Private Static Attributes

QMap<int, std::shared\_ptr<*sh::filesystem::DetailColumn*>> \_detailColumnsMap

QList<std::shared\_ptr<*sh::filesystem::DetailColumn*>> \_detailColumns

QMutex \_detailColumnsMutex

### Friends

**friend class** FilesystemOperationTransfers

**class** MyHandlerTransfer : **public** *sh::filesystem::FilesystemHandler::HandlerTransfer*

### Public Functions

**MyHandlerTransfer** (*sh::filesystem::FilesystemOperationProgressMonitor* \*progressmon = 0)

**~MyHandlerTransfer** ()

**void** respectCancel ()

**void** incrementTransferredBytes (qint64 donebytes)

### Private Members

*sh::filesystem::FilesystemOperationProgressMonitor* \*\_progressmon

## 10.1.166 Class *sh::filesystem::FilesystemOperationProgressMonitor*

**class** *sh::filesystem::FilesystemOperationProgressMonitor* : **public** std::enable\_shared\_from\_this<*FilesystemOperationProgressMonitor*>  
Implement this class and use an instance of it as parameter in some methods of *sh::filesystem::FilesystemOperation* for some additional functionality.

This includes monitoring the progress, specifying a resolution when conflicts in the filesystem would occur and more.

It also makes sense to directly instantiate this class in some cases.

Subclassed by *sh::actions::common::ActionAbstractTransferTree::MyFilesystemOperationProgressMonitor*,  
*sh::scripting::api::ApiFilesystemOperationProgressMonitor*

## Public Functions

**FilesystemOperationProgressMonitor** (*sh::actions::ActionExecutionInfo* \*actionExecution = 0)

Is intended to be directly constructed from everywhere.

### Parameters

- *actionExecution*: An optional *sh::actions::ActionExecutionInfo* which is used in some places in the default implementation, when available.

bool **hasItemInfo** ()

Checks if this progress monitor provides information about how many items of a certain total number are transferred so far.

quint64 **doneItems** ()

Checks how many items are transferred so far.

quint64 **allItems** ()

Checks how many items are to be transferred in total (as predicted in the current moment).

bool **hasBytesInfo** ()

Checks if this progress monitor provides information about how many byte of a certain total number are transferred so far.

quint64 **doneBytes** ()

Checks how many bytes are transferred so far.

quint64 **allBytes** ()

Checks how many bytes are to be transferred in total (as predicted in the current moment).

QString **getItemInfoFrom** ()

Returns the current source of transfer (as textual information).

QString **getItemInfoTo** ()

Returns the current destination of transfer (as textual information).

QString **estimation** ()

Returns the current performance and time estimation (as textual information).

**~FilesystemOperationProgressMonitor** ()

## Private Functions

void **setProgress** (quint64 *doneitems*, quint64 *allitems*, quint64 *donebytes*, quint64 *allbytes*)

Used by *FilesystemOperationTransfers* for setting status changes.

void **incProgress** (quint64 *doneitems*, quint64 *allitems*, quint64 *donebytes*, quint64 *allbytes*)

Used by *FilesystemOperationTransfers* for setting status changes.

void **setItemInfo** (QString *from*, QString *to*)

Used by *FilesystemOperationTransfers* for setting status changes.

void **setEstimation** (QString *estimation*)

Used by *FilesystemOperationTransfers* for setting status changes.

void **\_enablestatistics** ()

void **\_computestatistics** ()

void **\_triggerchanged** ()

```
void _stoptriggerchanged ()
```

### Private Members

```
QDateTime _laststatistictime
quint64 _laststatisticdonebytes = 0
quint64 _laststatisticdoneitems = 0
double _statisticbytespeed = 0.0
double _statisticitemspeed = 0.0
int _statisticcountdown = 4
QMutex _mutex
QMutex _triggermutex
quint64 _allitems = 0
quint64 _doneitems = 0
quint64 _allbytes = 0
quint64 _donebytes = 0
bool _triggerchanged_changedrunning = false
bool _triggerchanged_changedrunagain = false
bool _triggerchanged_stopped = false
QString _itemfrom
QString _itemto
QString _estimation
QList<std::shared_ptr<const sh::filesystem::Eurl>> changes
bool _begancomputestatistics = false
```

### Friends

```
friend class FilesystemOperationTransfers
friend class FilesystemOperation
```

## 10.1.167 Class *sh::filesystem::FilesystemOperationTransfers*

```
class sh::filesystem::FilesystemOperationTransfers
```

Used by *FilesystemOperation* for managing the execution of transfer operations.

This class is only used directly by *FilesystemOperation* (but some inner class might be interesting).

## Public Functions

**FilesystemOperationTransfers** (*FilesystemOperation* *\*filesystem*,  
std::shared\_ptr<*FilesystemOperationProgressMonitor*>  
*progressmon*)

Constructed only by the infrastructure and made available otherwise.

void **executestepqueue** ()

Executes the queue.

The execution model is as follows: The complete queue of *OperationStepClass* steps becomes expanded, filesystem conflicts gets resolved and then all steps become executed.

In detail:

At first there is an interleaved process of expanding all steps (see *OperationStepClass::expand*) and checking/resolving filesystem conflicts in the fresh tree (see *doreview*). It checks all steps in the queue for conflicts (e.g. destination already exists). It expands each step which has no conflicts (by enqueueing it to the stepqueue). After those checks, it asks *FilesystemOperationProgressMonitor::resolveConflicts* for a resolution for all outstanding conflicts. With all the steps, which were (or are) in conflict, the entire process becomes repeated. This ends when all steps are expanded and no open conflicts exist anymore.

When this process is done, the execution phase begins. It iterates the queue. It takes the (in FIFO manner) first element (if one is there, otherwise stops). For that step, it makes a late-time conflict check at first. If a conflict appears, it becomes temporarily unexpanded. A resolution loop as above will run for this element, eventually leading to re-expansion. After the check, when no conflicts are open, it executes the step.

void **addMoveItem** (std::shared\_ptr<const *sh::filesystem::Eurl*> *src*, std::shared\_ptr<const *sh::filesystem::Eurl*> *dest*)  
adds one move transfer into the queue.

void **addCopyItem** (std::shared\_ptr<const *sh::filesystem::Eurl*> *src*, std::shared\_ptr<const *sh::filesystem::Eurl*> *dest*)  
adds one copy transfer into the queue.

**~FilesystemOperationTransfers** ()

## Private Functions

void **\_compile\_moveItem** (std::shared\_ptr<const *sh::filesystem::Eurl*> *src*, std::shared\_ptr<const *sh::filesystem::Eurl*> *dest*, QList<*OperationStepClass*\*> *\*result*, QList<*OperationStepClass*\*> *withExpansion*)

Adds a move transfer step into given list.

void **\_compile\_copyItem** (std::shared\_ptr<const *sh::filesystem::Eurl*> *src*, std::shared\_ptr<const *sh::filesystem::Eurl*> *dest*, QList<*OperationStepClass*\*> *\*result*, QList<*OperationStepClass*\*> *withExpansion*)

Adds a copy transfer step into given list.

void **computeProgress** ()

Computes the current progress (how many items/bytes are transferred from which total?) and notifies the progress monitor.

bool **doreview** (QList<*OperationStepClass*\*> *reviewsteps*)

Makes checks if the given queue has conflicts and for asks *resolveConflicts* how to handle them.

For each non-conflicting step, it triggers its expansion, thereby modifying the stepqueue.

It returns when no open conflicts exist.

**Return** If there were any conflicts.

## Private Members

`QList<OperationStepClass*> stepqueue`

The queue of steps for execution.

`OperationStepClass *currentstep = 0`

The step which is currently in execution (or 0).

`QList<OperationStepClass*> donesteps`

Steps which were already executed.

`FilesystemOperation *filesystem`

`std::shared_ptr<FilesystemOperationProgressMonitor> progressmon`

**class OperationStep**

One step of execution in a transfer operation by *FilesystemOperation*.

It is typically about a certain source and destination in the filesystem. It also can stay in conflict (due to checks from *FilesystemOperationTransfers::doreview*) and provides methods for resolving them (in *FilesystemOperationTransfers::FilesystemOperationProgressMonitor::resolveConflicts*).

Different subclasses implement distinct kinds of operations (e.g. copying or deleting).

Subclassed by *sh::filesystem::FilesystemOperationTransfers::OperationStepClass*

## Public Types

**enum ConflictResolution**

Enumeration of ways how to resolve a filesystem conflict.

*Values:*

**enumerator Skip**

Skip this element.

**enumerator OverwriteDestination**

Overwrite the destination.

**enumerator RenameDestinationBefore**

Rename the file at the destination before transferring.

**enumerator UseDifferentDestinationName**

Transfer to another destination filename.

**enumerator MergeDirectories**

Merge source into destination directory (recursively).

**enumerator Unresolved**

No strategy.

**enumerator Indirect**

Indirectly solved. Only set by the engine.



## Public Functions

**OperationStep** (*FilesystemOperationTransfers* \*transfers, std::shared\_ptr<const sh::filesystem::Eurl> source, std::shared\_ptr<const sh::filesystem::Eurl> destination)  
Constructed only by the infrastructure and made available otherwise.

QString **conflictDescription** ()  
The description of the conflict (if any) as text.

*ConflictResolution* **conflictResolution** ()  
The currently chosen conflict resolution.

QString **conflictResolution\_renameDestinationBeforeTo** ()  
Returns new destination name (if current conflict resolution is *ConflictResolution::RenameDestinationBefore*).

QString **conflictResolution\_differentDestinationNameTo** ()  
Returns new destination name (if current conflict resolution is *ConflictResolution::UseDifferentDestinationName*).

std::shared\_ptr<const sh::filesystem::Eurl> **source** ()  
Returns the source location to be transferred (if specified).

std::shared\_ptr<const sh::filesystem::Eurl> **originalDestination** ()  
Returns the destination location (if specified).  
  
This is the complete target path, never the parent directory of the new element.  
  
This is the original destination location. See also effectiveDestination.

std::shared\_ptr<const sh::filesystem::Eurl> **effectiveDestination** ()  
Returns the real destination location with conflict resolution applied.

int **sourcetype** ()  
Returns the node type of the source.  
  
Should be overwritten in subclasses whenever it can determine the source type faster than the infrastructure.

void **setConflictResolve\_Skip** ()  
Set conflict resolution to *ConflictResolution::Skip*.

void **setConflictResolve\_OverwriteDestination** ()  
Set conflict resolution to *ConflictResolution::OverwriteDestination*.

void **setConflictResolve\_RenameDestinationBefore** (QString newname)  
Set conflict resolution to *ConflictResolution::RenameDestinationBefore*.

void **setConflictResolve\_UseDifferentDestinationName** (QString newname)  
Set conflict resolution to *ConflictResolution::UseDifferentDestinationName*.

void **setConflictResolve\_MergeDirectories** ()  
Set conflict resolution to *ConflictResolution::MergeDirectories*.  
  
See also setConflictResolve\_MergeDirectories\_isAllowed.

bool **setConflictResolve\_MergeDirectories\_checkAllowed** ()  
Checks if conflict resolution *ConflictResolution::MergeDirectories* is allowed.

~**OperationStep** ()

## Friends

```
friend class FilesystemOperationTransfers  
class OperationStep_ApplyDirectoryDetails : public sh::filesystem::FilesystemOperationTransfers::Operation
```

## Public Types

### enum ConflictResolution

Enumeration of ways how to resolve a filesystem conflict.

*Values:*

#### enumerator Skip

Skip this element.

#### enumerator OverwriteDestination

Overwrite the destination.

#### enumerator RenameDestinationBefore

Rename the file at the destination before transferring.

#### enumerator UseDifferentDestinationName

Transfer to another destination filename.

#### enumerator MergeDirectories

Merge source into destination directory (recursively).

#### enumerator Unresolved

No strategy.

#### enumerator Indirect

Indirectly solved. Only set by the engine.

## Public Functions

```
OperationStep_ApplyDirectoryDetails (FilesystemOperationTransfers  
                                     *transfers,          std::shared_ptr<const  
                                     sh::filesystem::Eurl>          item,  
                                     QList<QPair<sh::filesystem::DetailColumn*,  
                                     QVariant>>          values,  
                                     QList<OperationStepClass*>      withExpansion)  
                                     withExpansion)
```

```
void execute (Operation *op)
```

Implement this and handle the execution part of this step here.

```
bool checkconflicts (Operation *op)
```

Implement this for customizing the conflict checking.

```
QList<OperationStepClass*> expand ()
```

Implement this and handle the expansion part of this step here.

```
bool hasOwnProgressIncreaseHandling ()
```

If it takes care on its own to signal increases in the transfer progress.

```
void _unexpand (QList<OperationStepClass*> *intlist)
```

Bookkeeping in *FilesystemOperationTransfers*.

void **\_expand** (QList<OperationStepClass\*> \*intlist)  
 Bookkeeping in *FilesystemOperationTransfers*.

QList<QPair<sh::filesystem::DetailColumn\*, QVariant>> **getFileDetails** (sh::filesystem::Operation \*op,  
 std::shared\_ptr<const sh::filesystem::Eurl>  
 source)

void **applyFileDetails** (sh::filesystem::Operation \*op, QList<QPair<sh::filesystem::DetailColumn\*,  
 QVariant>> values, std::shared\_ptr<const sh::filesystem::Eurl> destination)

QString **conflictDescription** ()  
 The description of the conflict (if any) as text.

*ConflictResolution* **conflictResolution** ()  
 The currently chosen conflict resolution.

QString **conflictResolution\_renameDestinationBeforeTo** ()  
 Returns new destination name (if current conflict resolution is *ConflictResolution::RenameDestinationBefore*).

QString **conflictResolution\_differentDestinationNameTo** ()  
 Returns new destination name (if current conflict resolution is *ConflictResolution::UseDifferentDestinationName*).

std::shared\_ptr<const sh::filesystem::Eurl> **source** ()  
 Returns the source location to be transferred (if specified).

std::shared\_ptr<const sh::filesystem::Eurl> **originalDestination** ()  
 Returns the destination location (if specified).

This is the complete target path, never the parent directory of the new element.

This is the original destination location. See also *effectiveDestination*.

std::shared\_ptr<const sh::filesystem::Eurl> **effectiveDestination** ()  
 Returns the real destination location with conflict resolution applied.

int **sourcetype** ()  
 Returns the node type of the source.

Should be overwritten in subclasses whenever it can determine the source type faster than the infrastructure.

void **setConflictResolve\_Skip** ()  
 Set conflict resolution to *ConflictResolution::Skip*.

void **setConflictResolve\_OverwriteDestination** ()  
 Set conflict resolution to *ConflictResolution::OverwriteDestination*.

void **setConflictResolve\_RenameDestinationBefore** (QString newname)  
 Set conflict resolution to *ConflictResolution::RenameDestinationBefore*.

void **setConflictResolve\_UseDifferentDestinationName** (QString newname)  
 Set conflict resolution to *ConflictResolution::UseDifferentDestinationName*.

void **setConflictResolve\_MergeDirectories** ()  
 Set conflict resolution to *ConflictResolution::MergeDirectories*.

See also *setConflictResolve\_MergeDirectories\_isAllowed*.

bool **setConflictResolve\_MergeDirectories\_checkAllowed** ()  
Checks if conflict resolution *ConflictResolution::MergeDirectories* is allowed.

## Public Members

QList<QPair<*sh::filesystem::DetailColumn*\*, QVariant>> **\_detailvals**

quint64 **\_cntItems** = 0  
Number of items to be transferred in this step (used for progress monitoring).

quint64 **\_cntBytes** = 0  
Number of byte to be transferred in this step (used for progress monitoring).

bool **\_isExpanded** = false  
Bookkeeping in *FilesystemOperationTransfers*.

QList<*OperationStepClass*\*> **\_expandnodes**  
Bookkeeping in *FilesystemOperationTransfers*.

*OperationStepClass* \* **\_parentnode** = 0  
Bookkeeping in *FilesystemOperationTransfers*.

bool **\_deleteonunexpand** = true  
Bookkeeping in *FilesystemOperationTransfers*.

QList<*OperationStepClass*\*> **\_withExpansion**

**class OperationStep\_CopyDirectory** : public *sh::filesystem::FilesystemOperationTransfers::OperationStepClass*

## Public Types

**enum ConflictResolution**  
Enumeration of ways how to resolve a filesystem conflict.

*Values:*

**enumerator Skip**  
Skip this element.

**enumerator OverwriteDestination**  
Overwrite the destination.

**enumerator RenameDestinationBefore**  
Rename the file at the destination before transferring.

**enumerator UseDifferentDestinationName**  
Transfer to another destination filename.

**enumerator MergeDirectories**  
Merge source into destination directory (recursively).

**enumerator Unresolved**  
No strategy.

**enumerator Indirect**  
Indirectly solved. Only set by the engine.

## Public Functions

**OperationStep\_CopyDirectory** (*FilesystemOperationTransfers* *\*transfers*,  
 std::shared\_ptr<const *sh::filesystem::Eurl*> *src*,  
 std::shared\_ptr<const *sh::filesystem::Eurl*> *dest*,  
 QList<*OperationStepClass*\*> *withExpansion*)

void **execute** (*Operation* \**op*)  
 Implement this and handle the execution part of this step here.

QList<*OperationStepClass*\*> **expand** ()  
 Implement this and handle the expansion part of this step here.

int **sourcetype** ()  
 Returns the node type of the source.  
 Should be overwritten in subclasses whenever it can determine the source type faster than the infrastructure.

bool **checkconflicts** (*Operation* \**op*)  
 Implement this for customizing the conflict checking.

bool **hasOwnProgressIncreaseHandling** ()  
 If it takes care on its own to signal increases in the transfer progress.

void **\_unexpand** (QList<*OperationStepClass*\*> \**intlist*)  
 Bookkeeping in *FilesystemOperationTransfers*.

void **\_expand** (QList<*OperationStepClass*\*> \**intlist*)  
 Bookkeeping in *FilesystemOperationTransfers*.

QList<QPair<*sh::filesystem::DetailColumn*\*, QVariant>> **getFileDetails** (*sh::filesystem::Operation* \**op*,  
 std::shared\_ptr<const *sh::filesystem::Eurl*> *source*)

void **applyFileDetails** (*sh::filesystem::Operation* \**op*, QList<QPair<*sh::filesystem::DetailColumn*\*, QVariant>> *values*, std::shared\_ptr<const *sh::filesystem::Eurl*> *destination*)

QString **conflictDescription** ()  
 The description of the conflict (if any) as text.

*ConflictResolution* **conflictResolution** ()  
 The currently chosen conflict resolution.

QString **conflictResolution\_renameDestinationBeforeTo** ()  
 Returns new destination name (if current conflict resolution is *ConflictResolution::RenameDestinationBefore*).

QString **conflictResolution\_differentDestinationNameTo** ()  
 Returns new destination name (if current conflict resolution is *ConflictResolution::UseDifferentDestinationName*).

std::shared\_ptr<const *sh::filesystem::Eurl*> **source** ()  
 Returns the source location to be transferred (if specified).

std::shared\_ptr<const *sh::filesystem::Eurl*> **originalDestination** ()  
 Returns the destination location (if specified).  
 This is the complete target path, never the parent directory of the new element.  
 This is the original destination location. See also *effectiveDestination*.

```
std::shared_ptr<const sh::filesystem::Eurl> effectiveDestination ()
    Returns the real destination location with conflict resolution applied.

void setConflictResolve_Skip ()
    Set conflict resolution to ConflictResolution::Skip.

void setConflictResolve_OverwriteDestination ()
    Set conflict resolution to ConflictResolution::OverwriteDestination.

void setConflictResolve_RenameDestinationBefore (QString newname)
    Set conflict resolution to ConflictResolution::RenameDestinationBefore.

void setConflictResolve_UseDifferentDestinationName (QString newname)
    Set conflict resolution to ConflictResolution::UseDifferentDestinationName.

void setConflictResolve_MergeDirectories ()
    Set conflict resolution to ConflictResolution::MergeDirectories.

    See also setConflictResolve_MergeDirectories_isAllowed.

bool setConflictResolve_MergeDirectories_checkAllowed ()
    Checks if conflict resolution ConflictResolution::MergeDirectories is allowed.
```

## Public Members

```
quint64 _cntItems = 0
    Number of items to be transferred in this step (used for progress monitoring).

quint64 _cntBytes = 0
    Number of byte to be transferred in this step (used for progress monitoring).

bool _isExpanded = false
    Bookkeeping in FilesystemOperationTransfers.

QList<OperationStepClass*> _expandnodes
    Bookkeeping in FilesystemOperationTransfers.

OperationStepClass *_parentnode = 0
    Bookkeeping in FilesystemOperationTransfers.

bool _deleteonunexpand = true
    Bookkeeping in FilesystemOperationTransfers.

QList<OperationStepClass*> _withExpansion

class OperationStep_CopyFile : public sh::filesystem::FilesystemOperationTransfers::OperationStepClass
```

## Public Types

```
enum ConflictResolution
    Enumeration of ways how to resolve a filesystem conflict.

    Values:

    enumerator Skip
        Skip this element.

    enumerator OverwriteDestination
        Overwrite the destination.

    enumerator RenameDestinationBefore
        Rename the file at the destination before transferring.
```

**enumerator UseDifferentDestinationName**

Transfer to another destination filename.

**enumerator MergeDirectories**

Merge source into destination directory (recursively).

**enumerator Unresolved**

No strategy.

**enumerator Indirect**

Indirectly solved. Only set by the engine.

## Public Functions

**OperationStep\_CopyFile** (*FilesystemOperationTransfers* \*transfers, std::shared\_ptr<const sh::filesystem::Eurl> src, std::shared\_ptr<const sh::filesystem::Eurl> dest, QList<OperationStepClass\*> withExpansion)

void **execute** (*Operation* \*op)

Implement this and handle the execution part of this step here.

int **sourcetype** ()

Returns the node type of the source.

Should be overwritten in subclasses whenever it can determine the source type faster than the infrastructure.

bool **hasOwnProgressIncreaseHandling** ()

If it takes care on its own to signal increases in the transfer progress.

QList<OperationStepClass\*> **expand** ()

Implement this and handle the expansion part of this step here.

bool **checkconflicts** (*Operation* \*op)

Implement this for customizing the conflict checking.

void **\_unexpand** (QList<OperationStepClass\*> \*intlist)

Bookkeeping in *FilesystemOperationTransfers*.

void **\_expand** (QList<OperationStepClass\*> \*intlist)

Bookkeeping in *FilesystemOperationTransfers*.

QList<QPair<sh::filesystem::DetailColumn\*, QVariant>> **getFileDetails** (sh::filesystem::Operation \*op, std::shared\_ptr<const sh::filesystem::Eurl> source)

void **applyFileDetails** (sh::filesystem::Operation \*op, QList<QPair<sh::filesystem::DetailColumn\*, QVariant>> values, std::shared\_ptr<const sh::filesystem::Eurl> destination)

QString **conflictDescription** ()

The description of the conflict (if any) as text.

*ConflictResolution* **conflictResolution** ()

The currently chosen conflict resolution.

QString **conflictResolution\_renameDestinationBeforeTo** ()

Returns new destination name (if current conflict resolution is *ConflictResolution::RenameDestinationBefore*).

QString **conflictResolution\_differentDestinationNameTo** ()  
Returns new destination name (if current conflict resolution is *ConflictResolution::UseDifferentDestinationName*).

std::shared\_ptr<const *sh::filesystem::Eurl*> **source** ()  
Returns the source location to be transferred (if specified).

std::shared\_ptr<const *sh::filesystem::Eurl*> **originalDestination** ()  
Returns the destination location (if specified).  
  
This is the complete target path, never the parent directory of the new element.  
  
This is the original destination location. See also *effectiveDestination*.

std::shared\_ptr<const *sh::filesystem::Eurl*> **effectiveDestination** ()  
Returns the real destination location with conflict resolution applied.

void **setConflictResolve\_Skip** ()  
Set conflict resolution to *ConflictResolution::Skip*.

void **setConflictResolve\_OverwriteDestination** ()  
Set conflict resolution to *ConflictResolution::OverwriteDestination*.

void **setConflictResolve\_RenameDestinationBefore** (QString *newname*)  
Set conflict resolution to *ConflictResolution::RenameDestinationBefore*.

void **setConflictResolve\_UseDifferentDestinationName** (QString *newname*)  
Set conflict resolution to *ConflictResolution::UseDifferentDestinationName*.

void **setConflictResolve\_MergeDirectories** ()  
Set conflict resolution to *ConflictResolution::MergeDirectories*.  
  
See also *setConflictResolve\_MergeDirectories\_isAllowed*.

bool **setConflictResolve\_MergeDirectories\_checkAllowed** ()  
Checks if conflict resolution *ConflictResolution::MergeDirectories* is allowed.

## Public Members

quint64 **\_cntItems** = 0  
Number of items to be transferred in this step (used for progress monitoring).

quint64 **\_cntBytes** = 0  
Number of byte to be transferred in this step (used for progress monitoring).

bool **\_isExpanded** = false  
Bookkeeping in *FilesystemOperationTransfers*.

QList<OperationStepClass\*> **\_expandnodes**  
Bookkeeping in *FilesystemOperationTransfers*.

OperationStepClass \* **\_parentnode** = 0  
Bookkeeping in *FilesystemOperationTransfers*.

bool **\_deleteonunexpand** = true  
Bookkeeping in *FilesystemOperationTransfers*.

QList<OperationStepClass\*> **\_withExpansion**

**class OperationStep\_CopyLink : public *sh::filesystem::FilesystemOperationTransfers::OperationStepClass***



## Public Types

### enum ConflictResolution

Enumeration of ways how to resolve a filesystem conflict.

*Values:*

#### enumerator Skip

Skip this element.

#### enumerator OverwriteDestination

Overwrite the destination.

#### enumerator RenameDestinationBefore

Rename the file at the destination before transferring.

#### enumerator UseDifferentDestinationName

Transfer to another destination filename.

#### enumerator MergeDirectories

Merge source into destination directory (recursively).

#### enumerator Unresolved

No strategy.

#### enumerator Indirect

Indirectly solved. Only set by the engine.

## Public Functions

**OperationStep\_CopyLink** (*FilesystemOperationTransfers* \*transfers, std::shared\_ptr<const sh::filesystem::Eurl> src, std::shared\_ptr<const sh::filesystem::Eurl> dest, QList<OperationStepClass\*> withExpansion)

void **execute** (*Operation* \*op)

Implement this and handle the execution part of this step here.

int **sourcetype** ()

Returns the node type of the source.

Should be overwritten in subclasses whenever it can determine the source type faster than the infrastructure.

QList<OperationStepClass\*> **expand** ()

Implement this and handle the expansion part of this step here.

bool **checkconflicts** (*Operation* \*op)

Implement this for customizing the conflict checking.

bool **hasOwnProgressIncreaseHandling** ()

If it takes care on its own to signal increases in the transfer progress.

void **\_unexpand** (QList<OperationStepClass\*> \*intlist)

Bookkeeping in *FilesystemOperationTransfers*.

void **\_expand** (QList<OperationStepClass\*> \*intlist)

Bookkeeping in *FilesystemOperationTransfers*.

`QList<QPair<sh::filesystem::DetailColumn*, QVariant>> getFileDetails (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> source)`

`void applyFileDetails (sh::filesystem::Operation *op, QList<QPair<sh::filesystem::DetailColumn*, QVariant>> values, std::shared_ptr<const sh::filesystem::Eurl> destination)`

`QString conflictDescription ()`  
The description of the conflict (if any) as text.

`ConflictResolution conflictResolution ()`  
The currently chosen conflict resolution.

`QString conflictResolution_renameDestinationBeforeTo ()`  
Returns new destination name (if current conflict resolution is *ConflictResolution::RenameDestinationBefore*).

`QString conflictResolution_differentDestinationNameTo ()`  
Returns new destination name (if current conflict resolution is *ConflictResolution::UseDifferentDestinationName*).

`std::shared_ptr<const sh::filesystem::Eurl> source ()`  
Returns the source location to be transferred (if specified).

`std::shared_ptr<const sh::filesystem::Eurl> originalDestination ()`  
Returns the destination location (if specified).

This is the complete target path, never the parent directory of the new element.

This is the original destination location. See also `effectiveDestination`.

`std::shared_ptr<const sh::filesystem::Eurl> effectiveDestination ()`  
Returns the real destination location with conflict resolution applied.

`void setConflictResolve_Skip ()`  
Set conflict resolution to *ConflictResolution::Skip*.

`void setConflictResolve_OverwriteDestination ()`  
Set conflict resolution to *ConflictResolution::OverwriteDestination*.

`void setConflictResolve_RenameDestinationBefore (QString newname)`  
Set conflict resolution to *ConflictResolution::RenameDestinationBefore*.

`void setConflictResolve_UseDifferentDestinationName (QString newname)`  
Set conflict resolution to *ConflictResolution::UseDifferentDestinationName*.

`void setConflictResolve_MergeDirectories ()`  
Set conflict resolution to *ConflictResolution::MergeDirectories*.

See also `setConflictResolve_MergeDirectories_isAllowed`.

`bool setConflictResolve_MergeDirectories_checkAllowed ()`  
Checks if conflict resolution *ConflictResolution::MergeDirectories* is allowed.

## Public Members

```

quint64 _cntItems = 0
    Number of items to be transferred in this step (used for progress monitoring).

quint64 _cntBytes = 0
    Number of byte to be transferred in this step (used for progress monitoring).

bool _isExpanded = false
    Bookkeeping in FilesystemOperationTransfers.

QList<OperationStepClass*> _expandnodes
    Bookkeeping in FilesystemOperationTransfers.

OperationStepClass * _parentnode = 0
    Bookkeeping in FilesystemOperationTransfers.

bool _deleteonunexpand = true
    Bookkeeping in FilesystemOperationTransfers.

QList<OperationStepClass*> _withExpansion

class OperationStep_DeleteItem: public sh::filesystem::FilesystemOperationTransfers::OperationStepClass

```

## Public Types

```

enum ConflictResolution
    Enumeration of ways how to resolve a filesystem conflict.

    Values:

    enumerator Skip
        Skip this element.

    enumerator OverwriteDestination
        Overwrite the destination.

    enumerator RenameDestinationBefore
        Rename the file at the destination before transferring.

    enumerator UseDifferentDestinationName
        Transfer to another destination filename.

    enumerator MergeDirectories
        Merge source into destination directory (recursively).

    enumerator Unresolved
        No strategy.

    enumerator Indirect
        Indirectly solved. Only set by the engine.

```

## Public Functions

**OperationStep\_DeleteItem** (*FilesystemOperationTransfers* *\*transfers*,  
std::shared\_ptr<const *sh::filesystem::Eurl*> *src*,  
std::shared\_ptr<const *sh::filesystem::Eurl*> *dest*,  
QList<*OperationStepClass*\*> *withExpansion*)

void **execute** (*Operation* \**op*)  
Implement this and handle the execution part of this step here.

bool **checkconflicts** (*Operation* \**op*)  
Implement this for customizing the conflict checking.

int **sourcetype** ()  
Returns the node type of the source.  
  
Should be overwritten in subclasses whenever it can determine the source type faster than the infrastructure.

QList<*OperationStepClass*\*> **expand** ()  
Implement this and handle the expansion part of this step here.

bool **hasOwnProgressIncreaseHandling** ()  
If it takes care on its own to signal increases in the transfer progress.

void **\_unexpand** (QList<*OperationStepClass*\*> \**intlist*)  
Bookkeeping in *FilesystemOperationTransfers*.

void **\_expand** (QList<*OperationStepClass*\*> \**intlist*)  
Bookkeeping in *FilesystemOperationTransfers*.

QList<QPair<*sh::filesystem::DetailColumn*\*, QVariant>> **getFileDetails** (*sh::filesystem::Operation* \**op*,  
std::shared\_ptr<const *sh::filesystem::Eurl*> *source*)

void **applyFileDetails** (*sh::filesystem::Operation* \**op*, QList<QPair<*sh::filesystem::DetailColumn*\*,  
QVariant>> *values*, std::shared\_ptr<const *sh::filesystem::Eurl*> *destination*)

QString **conflictDescription** ()  
The description of the conflict (if any) as text.

*ConflictResolution* **conflictResolution** ()  
The currently chosen conflict resolution.

QString **conflictResolution\_renameDestinationBeforeTo** ()  
Returns new destination name (if current conflict resolution is *ConflictResolution::RenameDestinationBefore*).

QString **conflictResolution\_differentDestinationNameTo** ()  
Returns new destination name (if current conflict resolution is *ConflictResolution::UseDifferentDestinationName*).

std::shared\_ptr<const *sh::filesystem::Eurl*> **source** ()  
Returns the source location to be transferred (if specified).

std::shared\_ptr<const *sh::filesystem::Eurl*> **originalDestination** ()  
Returns the destination location (if specified).  
  
This is the complete target path, never the parent directory of the new element.  
  
This is the original destination location. See also *effectiveDestination*.

```
std::shared_ptr<const sh::filesystem::Eurl> effectiveDestination ()
    Returns the real destination location with conflict resolution applied.

void setConflictResolve_Skip ()
    Set conflict resolution to ConflictResolution::Skip.

void setConflictResolve_OverwriteDestination ()
    Set conflict resolution to ConflictResolution::OverwriteDestination.

void setConflictResolve_RenameDestinationBefore (QString newname)
    Set conflict resolution to ConflictResolution::RenameDestinationBefore.

void setConflictResolve_UseDifferentDestinationName (QString newname)
    Set conflict resolution to ConflictResolution::UseDifferentDestinationName.

void setConflictResolve_MergeDirectories ()
    Set conflict resolution to ConflictResolution::MergeDirectories.

    See also setConflictResolve_MergeDirectories_isAllowed.

bool setConflictResolve_MergeDirectories_checkAllowed ()
    Checks if conflict resolution ConflictResolution::MergeDirectories is allowed.
```

## Public Members

```
quint64 _cntItems = 0
    Number of items to be transferred in this step (used for progress monitoring).

quint64 _cntBytes = 0
    Number of byte to be transferred in this step (used for progress monitoring).

bool _isExpanded = false
    Bookkeeping in FilesystemOperationTransfers.

QList<OperationStepClass*> _expandnodes
    Bookkeeping in FilesystemOperationTransfers.

OperationStepClass *_parentnode = 0
    Bookkeeping in FilesystemOperationTransfers.

bool _deleteonunexpand = true
    Bookkeeping in FilesystemOperationTransfers.

QList<OperationStepClass*> _withExpansion

class OperationStep_RenameItem: public sh::filesystem::FilesystemOperationTransfers::OperationStepClass
```

## Public Types

```
enum ConflictResolution
    Enumeration of ways how to resolve a filesystem conflict.

    Values:

    enumerator Skip
        Skip this element.

    enumerator OverwriteDestination
        Overwrite the destination.

    enumerator RenameDestinationBefore
        Rename the file at the destination before transferring.
```

**enumerator UseDifferentDestinationName**

Transfer to another destination filename.

**enumerator MergeDirectories**

Merge source into destination directory (recursively).

**enumerator Unresolved**

No strategy.

**enumerator Indirect**

Indirectly solved. Only set by the engine.

## Public Functions

**OperationStep\_RenameItem** (*FilesystemOperationTransfers* *\*transfers*,  
std::shared\_ptr<const *sh::filesystem::Eurl*> *src*, QString  
*destpath*, QList<*OperationStepClass*\*> *withExpansion*)

QList<*OperationStepClass*\*> **expand** ()

Implement this and handle the expansion part of this step here.

void **execute** (*Operation* \**op*)

Implement this and handle the execution part of this step here.

bool **checkconflicts** (*Operation* \**op*)

Implement this for customizing the conflict checking.

bool **hasOwnProgressIncreaseHandling** ()

If it takes care on its own to signal increases in the transfer progress.

void **\_unexpand** (QList<*OperationStepClass*\*> \**intlist*)

Bookkeeping in *FilesystemOperationTransfers*.

void **\_expand** (QList<*OperationStepClass*\*> \**intlist*)

Bookkeeping in *FilesystemOperationTransfers*.

QList<QPair<*sh::filesystem::DetailColumn*\*, QVariant>> **getFileDetails** (*sh::filesystem::Operation*  
\**op*,  
std::shared\_ptr<const  
*sh::filesystem::Eurl*>  
*source*)

void **applyFileDetails** (*sh::filesystem::Operation* \**op*, QList<QPair<*sh::filesystem::DetailColumn*\*,  
QVariant>> *values*, std::shared\_ptr<const *sh::filesystem::Eurl*> *des-*  
*tination*)

QString **conflictDescription** ()

The description of the conflict (if any) as text.

*ConflictResolution* **conflictResolution** ()

The currently chosen conflict resolution.

QString **conflictResolution\_renameDestinationBeforeTo** ()

Returns new destination name (if current conflict resolution is *ConflictResolu-*  
*tion::RenameDestinationBefore*).

QString **conflictResolution\_differentDestinationNameTo** ()

Returns new destination name (if current conflict resolution is *ConflictResolu-*  
*tion::UseDifferentDestinationName*).

std::shared\_ptr<const *sh::filesystem::Eurl*> **source** ()

Returns the source location to be transferred (if specified).

`std::shared_ptr<const sh::filesystem::Eurl> originalDestination ()`

Returns the destination location (if specified).

This is the complete target path, never the parent directory of the new element.

This is the original destination location. See also `effectiveDestination`.

`std::shared_ptr<const sh::filesystem::Eurl> effectiveDestination ()`

Returns the real destination location with conflict resolution applied.

`int sourcetype ()`

Returns the node type of the source.

Should be overwritten in subclasses whenever it can determine the source type faster than the infrastructure.

`void setConflictResolve_Skip ()`

Set conflict resolution to *ConflictResolution::Skip*.

`void setConflictResolve_OverwriteDestination ()`

Set conflict resolution to *ConflictResolution::OverwriteDestination*.

`void setConflictResolve_RenameDestinationBefore (QString newname)`

Set conflict resolution to *ConflictResolution::RenameDestinationBefore*.

`void setConflictResolve_UseDifferentDestinationName (QString newname)`

Set conflict resolution to *ConflictResolution::UseDifferentDestinationName*.

`void setConflictResolve_MergeDirectories ()`

Set conflict resolution to *ConflictResolution::MergeDirectories*.

See also `setConflictResolve_MergeDirectories_isAllowed`.

`bool setConflictResolve_MergeDirectories_checkAllowed ()`

Checks if conflict resolution *ConflictResolution::MergeDirectories* is allowed.

## Public Members

`quint64 _cntItems = 0`

Number of items to be transferred in this step (used for progress monitoring).

`quint64 _cntBytes = 0`

Number of byte to be transferred in this step (used for progress monitoring).

`bool _isExpanded = false`

Bookkeeping in *FilesystemOperationTransfers*.

`QList<OperationStepClass*> _expandnodes`

Bookkeeping in *FilesystemOperationTransfers*.

`OperationStepClass *_parentnode = 0`

Bookkeeping in *FilesystemOperationTransfers*.

`bool _deleteonunexpand = true`

Bookkeeping in *FilesystemOperationTransfers*.

`QList<OperationStepClass*> _withExpansion`

**class OperationStepClass: public *sh::filesystem::FilesystemOperationTransfers::OperationStep***

A base class for all implementations.

They are used from inside *FilesystemOperationTransfers::executestepqueue*.

Subclassed by *sh::filesystem::FilesystemOperationTransfers::OperationStep\_ApplyDirectoryDetails*,  
*sh::filesystem::FilesystemOperationTransfers::OperationStep\_CopyDirectory*,  
*sh::filesystem::FilesystemOperationTransfers::OperationStep\_CopyFile*, *sh::filesystem::FilesystemOperationTransfers::Op*  
*sh::filesystem::FilesystemOperationTransfers::OperationStep\_DeleteItem*,  
*sh::filesystem::FilesystemOperationTransfers::OperationStep\_RenameItem*

## Public Types

### enum ConflictResolution

Enumeration of ways how to resolve a filesystem conflict.

Values:

#### enumerator Skip

Skip this element.

#### enumerator OverwriteDestination

Overwrite the destination.

#### enumerator RenameDestinationBefore

Rename the file at the destination before transferring.

#### enumerator UseDifferentDestinationName

Transfer to another destination filename.

#### enumerator MergeDirectories

Merge source into destination directory (recursively).

#### enumerator Unresolved

No strategy.

#### enumerator Indirect

Indirectly solved. Only set by the engine.

## Public Functions

**OperationStepClass** (*FilesystemOperationTransfers* \*transfers, std::shared\_ptr<const  
*sh::filesystem::Eurl*> source, std::shared\_ptr<const *Eurl*> destination,  
QList<*OperationStepClass*> \*withExpansion)

void **execute** (*Operation* \*op) = 0

Implement this and handle the execution part of this step here.

QList<*OperationStepClass*> **expand** ()

Implement this and handle the expansion part of this step here.

bool **checkconflicts** (*Operation* \*op)

Implement this for customizing the conflict checking.

bool **hasOwnProgressIncreaseHandling** ()

If it takes care on its own to signal increases in the transfer progress.

void **\_unexpand** (QList<*OperationStepClass*> \*intlist)

Bookkeeping in *FilesystemOperationTransfers*.

void **\_expand** (QList<*OperationStepClass*> \*intlist)

Bookkeeping in *FilesystemOperationTransfers*.

**~OperationStepClass** ()



```

QList<QPair<sh::filesystem::DetailColumn*, QVariant>> getFileDetails (sh::filesystem::Operation
                                                                    *op,
                                                                    std::shared_ptr<const
                                                                    sh::filesystem::Eurl>
                                                                    source)

void applyFileDetails (sh::filesystem::Operation *op, QList<QPair<sh::filesystem::DetailColumn*,
                                                                    QVariant>> values, std::shared_ptr<const sh::filesystem::Eurl> des-
                                                                    tination)

QString conflictDescription ()
    The description of the conflict (if any) as text.

ConflictResolution conflictResolution ()
    The currently chosen conflict resolution.

QString conflictResolution_renameDestinationBeforeTo ()
    Returns new destination name (if current conflict resolution is ConflictResolu-
    tion::RenameDestinationBefore).

QString conflictResolution_differentDestinationNameTo ()
    Returns new destination name (if current conflict resolution is ConflictResolu-
    tion::UseDifferentDestinationName).

std::shared_ptr<const sh::filesystem::Eurl> source ()
    Returns the source location to be transferred (if specified).

std::shared_ptr<const sh::filesystem::Eurl> originalDestination ()
    Returns the destination location (if specified).

    This is the complete target path, never the parent directory of the new element.

    This is the original destination location. See also effectiveDestination.

std::shared_ptr<const sh::filesystem::Eurl> effectiveDestination ()
    Returns the real destination location with conflict resolution applied.

int sourcetype ()
    Returns the node type of the source.

    Should be overwritten in subclasses whenever it can determine the source type faster than the infras-
    tructure.

void setConflictResolve_Skip ()
    Set conflict resolution to ConflictResolution::Skip.

void setConflictResolve_OverwriteDestination ()
    Set conflict resolution to ConflictResolution::OverwriteDestination.

void setConflictResolve_RenameDestinationBefore (QString newname)
    Set conflict resolution to ConflictResolution::RenameDestinationBefore.

void setConflictResolve_UseDifferentDestinationName (QString newname)
    Set conflict resolution to ConflictResolution::UseDifferentDestinationName.

void setConflictResolve_MergeDirectories ()
    Set conflict resolution to ConflictResolution::MergeDirectories.

    See also setConflictResolve_MergeDirectories_isAllowed.

bool setConflictResolve_MergeDirectories_checkAllowed ()
    Checks if conflict resolution ConflictResolution::MergeDirectories is allowed.

```

### Public Members

quint64 **\_cntItems** = 0  
Number of items to be transferred in this step (used for progress monitoring).

quint64 **\_cntBytes** = 0  
Number of byte to be transferred in this step (used for progress monitoring).

bool **\_isExpanded** = false  
Bookkeeping in *FilesystemOperationTransfers*.

QList<*OperationStepClass*\*> **\_expandnodes**  
Bookkeeping in *FilesystemOperationTransfers*.

*OperationStepClass* \* **\_parentnode** = 0  
Bookkeeping in *FilesystemOperationTransfers*.

bool **\_deleteonunexpand** = true  
Bookkeeping in *FilesystemOperationTransfers*.

QList<*OperationStepClass*\*> **\_withExpansion**

**class SingleStepMonitor**: public *sh::filesystem::FilesystemHandler::HandlerTransfer*  
Used for managing detailed progress changes about a single step.

### Public Functions

**SingleStepMonitor** (*OperationStepClass* \*step)

void **respectCancel** ()

void **incrementTransferredBytes** (quint64 donebytes)

**~SingleStepMonitor** ()

### Private Members

*OperationStepClass* \* **step**

quint64 **\_donebytes** = 0

## 10.1.168 Class *sh::filesystem::FilesystemOperationTransfers::OperationStep*

**class** *sh::filesystem::FilesystemOperationTransfers::OperationStep*  
One step of execution in a transfer operation by *FilesystemOperation*.

It is typically about a certain source and destination in the filesystem. It also can stay in conflict (due to checks from *FilesystemOperationTransfers::doreview*) and provides methods for resolving them (in *FilesystemOperationTransfers::FilesystemOperationProgressMonitor::resolveConflicts*).

Different subclasses implement distinct kinds of operations (e.g. copying or deleting).

Subclassed by *sh::filesystem::FilesystemOperationTransfers::OperationStepClass*

## Public Types

### enum ConflictResolution

Enumeration of ways how to resolve a filesystem conflict.

*Values:*

#### enumerator Skip

Skip this element.

#### enumerator OverwriteDestination

Overwrite the destination.

#### enumerator RenameDestinationBefore

Rename the file at the destination before transferring.

#### enumerator UseDifferentDestinationName

Transfer to another destination filename.

#### enumerator MergeDirectories

Merge source into destination directory (recursively).

#### enumerator Unresolved

No strategy.

#### enumerator Indirect

Indirectly solved. Only set by the engine.

## Public Functions

**OperationStep** (*FilesystemOperationTransfers* \*transfers, std::shared\_ptr<const *sh::filesystem::Eurl*> source, std::shared\_ptr<const *sh::filesystem::Eurl*> destination)

Constructed only by the infrastructure and made available otherwise.

QString **conflictDescription** ()

The description of the conflict (if any) as text.

*ConflictResolution* **conflictResolution** ()

The currently chosen conflict resolution.

QString **conflictResolution\_renameDestinationBeforeTo** ()

Returns new destination name (if current conflict resolution is *ConflictResolution::RenameDestinationBefore*).

QString **conflictResolution\_differentDestinationNameTo** ()

Returns new destination name (if current conflict resolution is *ConflictResolution::UseDifferentDestinationName*).

std::shared\_ptr<const *sh::filesystem::Eurl*> **source** ()

Returns the source location to be transferred (if specified).

std::shared\_ptr<const *sh::filesystem::Eurl*> **originalDestination** ()

Returns the destination location (if specified).

This is the complete target path, never the parent directory of the new element.

This is the original destination location. See also effectiveDestination.

std::shared\_ptr<const *sh::filesystem::Eurl*> **effectiveDestination** ()

Returns the real destination location with conflict resolution applied.

**int sourcetype ()**  
Returns the node type of the source.  
  
Should be overwritten in subclasses whenever it can determine the source type faster than the infrastructure.

**void setConflictResolve\_Skip ()**  
Set conflict resolution to *ConflictResolution::Skip*.

**void setConflictResolve\_OverwriteDestination ()**  
Set conflict resolution to *ConflictResolution::OverwriteDestination*.

**void setConflictResolve\_RenameDestinationBefore (QString newname)**  
Set conflict resolution to *ConflictResolution::RenameDestinationBefore*.

**void setConflictResolve\_UseDifferentDestinationName (QString newname)**  
Set conflict resolution to *ConflictResolution::UseDifferentDestinationName*.

**void setConflictResolve\_MergeDirectories ()**  
Set conflict resolution to *ConflictResolution::MergeDirectories*.  
  
See also `setConflictResolve_MergeDirectories_isAllowed`.

**bool setConflictResolve\_MergeDirectories\_checkAllowed ()**  
Checks if conflict resolution *ConflictResolution::MergeDirectories* is allowed.

**~OperationStep ()**

## Friends

**friend class** FilesystemOperationTransfers

### 10.1.169 Class `sh::filesystem::FilesystemOperationTransfers::OperationStepClass`

**class** `sh::filesystem::FilesystemOperationTransfers::OperationStepClass` : **public** `sh::filesystem::File`

A base class for all implementations.

They are used from inside `FilesystemOperationTransfers::executestepqueue`.

Subclassed by `sh::filesystem::FilesystemOperationTransfers::OperationStep_ApplyDirectoryDetails`,  
`sh::filesystem::FilesystemOperationTransfers::OperationStep_CopyDirectory`, `sh::filesystem::FilesystemOperationTransfers::Op`  
`sh::filesystem::FilesystemOperationTransfers::OperationStep_CopyLink`, `sh::filesystem::FilesystemOperationTransfers::Operati`  
`sh::filesystem::FilesystemOperationTransfers::OperationStep_RenameItem`

## Public Types

**enum ConflictResolution**

Enumeration of ways how to resolve a filesystem conflict.

*Values:*

**enumerator Skip**

Skip this element.

**enumerator OverwriteDestination**

Overwrite the destination.

**enumerator RenameDestinationBefore**

Rename the file at the destination before transferring.

**enumerator UseDifferentDestinationName**

Transfer to another destination filename.

**enumerator MergeDirectories**

Merge source into destination directory (recursively).

**enumerator Unresolved**

No strategy.

**enumerator Indirect**

Indirectly solved. Only set by the engine.

## Public Functions

**OperationStepClass** (*FilesystemOperationTransfers* \*transfers, std::shared\_ptr<const *sh::filesystem::Eurl*> source, std::shared\_ptr<const *Eurl*> destination, QList<*OperationStepClass*\*> withExpansion)

void **execute** (*Operation* \*op) = 0

Implement this and handle the execution part of this step here.

QList<*OperationStepClass*\*> **expand** ()

Implement this and handle the expansion part of this step here.

bool **checkconflicts** (*Operation* \*op)

Implement this for customizing the conflict checking.

bool **hasOwnProgressIncreaseHandling** ()

If it takes care on its own to signal increases in the transfer progress.

void **\_unexpand** (QList<*OperationStepClass*\*> \*intlist)

Bookkeeping in *FilesystemOperationTransfers*.

void **\_expand** (QList<*OperationStepClass*\*> \*intlist)

Bookkeeping in *FilesystemOperationTransfers*.

**~OperationStepClass** ()

QList<QPair<*sh::filesystem::DetailColumn*\*, QVariant>> **getFileDetails** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> source)

void **applyFileDetails** (*sh::filesystem::Operation* \*op, QList<QPair<*sh::filesystem::DetailColumn*\*, QVariant>> values, std::shared\_ptr<const *sh::filesystem::Eurl*> destination)

QString **conflictDescription** ()

The description of the conflict (if any) as text.

*ConflictResolution* **conflictResolution** ()

The currently chosen conflict resolution.

QString **conflictResolution\_renameDestinationBeforeTo** ()

Returns new destination name (if current conflict resolution is *ConflictResolution::RenameDestinationBefore*).

QString **conflictResolution\_differentDestinationNameTo** ()

Returns new destination name (if current conflict resolution is *ConflictResolution::UseDifferentDestinationName*).

`std::shared_ptr<const sh::filesystem::Eurl> source ()`  
Returns the source location to be transferred (if specified).

`std::shared_ptr<const sh::filesystem::Eurl> originalDestination ()`  
Returns the destination location (if specified).

This is the complete target path, never the parent directory of the new element.

This is the original destination location. See also `effectiveDestination`.

`std::shared_ptr<const sh::filesystem::Eurl> effectiveDestination ()`  
Returns the real destination location with conflict resolution applied.

`int sourcetype ()`  
Returns the node type of the source.

Should be overwritten in subclasses whenever it can determine the source type faster than the infrastructure.

`void setConflictResolve_Skip ()`  
Set conflict resolution to *ConflictResolution::Skip*.

`void setConflictResolve_OverwriteDestination ()`  
Set conflict resolution to *ConflictResolution::OverwriteDestination*.

`void setConflictResolve_RenameDestinationBefore (QString newname)`  
Set conflict resolution to *ConflictResolution::RenameDestinationBefore*.

`void setConflictResolve_UseDifferentDestinationName (QString newname)`  
Set conflict resolution to *ConflictResolution::UseDifferentDestinationName*.

`void setConflictResolve_MergeDirectories ()`  
Set conflict resolution to *ConflictResolution::MergeDirectories*.

See also `setConflictResolve_MergeDirectories_isAllowed`.

`bool setConflictResolve_MergeDirectories_checkAllowed ()`  
Checks if conflict resolution *ConflictResolution::MergeDirectories* is allowed.

## Public Members

`quint64 _cntItems = 0`  
Number of items to be transferred in this step (used for progress monitoring).

`quint64 _cntBytes = 0`  
Number of byte to be transferred in this step (used for progress monitoring).

`bool _isExpanded = false`  
Bookkeeping in *FilesystemOperationTransfers*.

`QList<OperationStepClass*> _expandnodes`  
Bookkeeping in *FilesystemOperationTransfers*.

`OperationStepClass * _parentnode = 0`  
Bookkeeping in *FilesystemOperationTransfers*.

`bool _deleteonunexpand = true`  
Bookkeeping in *FilesystemOperationTransfers*.

`QList<OperationStepClass*> _withExpansion`

## 10.1.170 Class `sh::filesystem::FileSystemOperationTransfers::OperationStep_ApplyDirectoryDetails`

```
class sh::filesystem::FileSystemOperationTransfers::OperationStep_ApplyDirectoryDetails : public
```

### Public Types

#### enum `ConflictResolution`

Enumeration of ways how to resolve a filesystem conflict.

*Values:*

##### enumerator `Skip`

Skip this element.

##### enumerator `OverwriteDestination`

Overwrite the destination.

##### enumerator `RenameDestinationBefore`

Rename the file at the destination before transferring.

##### enumerator `UseDifferentDestinationName`

Transfer to another destination filename.

##### enumerator `MergeDirectories`

Merge source into destination directory (recursively).

##### enumerator `Unresolved`

No strategy.

##### enumerator `Indirect`

Indirectly solved. Only set by the engine.

### Public Functions

```
OperationStep_ApplyDirectoryDetails (FileSystemOperationTransfers *transfers,
                                     std::shared_ptr<const sh::filesystem::Eurl>
                                     item, QList<QPair<sh::filesystem::DetailColumn*,
                                     QVariant>> values, QList<OperationStepClass*>
                                     withExpansion)
```

```
void execute (Operation *op)
```

Implement this and handle the execution part of this step here.

```
bool checkconflicts (Operation *op)
```

Implement this for customizing the conflict checking.

```
QList<OperationStepClass*> expand ()
```

Implement this and handle the expansion part of this step here.

```
bool hasOwnProgressIncreaseHandling ()
```

If it takes care on its own to signal increases in the transfer progress.

```
void _unexpand (QList<OperationStepClass*> *intlist)
```

Bookkeeping in *FileSystemOperationTransfers*.

```
void _expand (QList<OperationStepClass*> *intlist)
```

Bookkeeping in *FileSystemOperationTransfers*.

`QList<QPair<sh::filesystem::DetailColumn*, QVariant>> getFileDetails (sh::filesystem::Operation *op,  
std::shared_ptr<const sh::filesystem::Eurl>  
source)`

`void applyFileDetails (sh::filesystem::Operation *op, QList<QPair<sh::filesystem::DetailColumn*,  
QVariant>> values, std::shared_ptr<const sh::filesystem::Eurl> destina-  
tion)`

`QString conflictDescription ()`  
The description of the conflict (if any) as text.

`ConflictResolution conflictResolution ()`  
The currently chosen conflict resolution.

`QString conflictResolution_renameDestinationBeforeTo ()`  
Returns new destination name (if current conflict resolution is *ConflictResolu-  
tion::RenameDestinationBefore*).

`QString conflictResolution_differentDestinationNameTo ()`  
Returns new destination name (if current conflict resolution is *ConflictResolu-  
tion::UseDifferentDestinationName*).

`std::shared_ptr<const sh::filesystem::Eurl> source ()`  
Returns the source location to be transferred (if specified).

`std::shared_ptr<const sh::filesystem::Eurl> originalDestination ()`  
Returns the destination location (if specified).

This is the complete target path, never the parent directory of the new element.

This is the original destination location. See also `effectiveDestination`.

`std::shared_ptr<const sh::filesystem::Eurl> effectiveDestination ()`  
Returns the real destination location with conflict resolution applied.

`int sourcetype ()`  
Returns the node type of the source.

Should be overwritten in subclasses whenever it can determine the source type faster than the infrastruc-  
ture.

`void setConflictResolve_Skip ()`  
Set conflict resolution to *ConflictResolution::Skip*.

`void setConflictResolve_OverwriteDestination ()`  
Set conflict resolution to *ConflictResolution::OverwriteDestination*.

`void setConflictResolve_RenameDestinationBefore (QString newname)`  
Set conflict resolution to *ConflictResolution::RenameDestinationBefore*.

`void setConflictResolve_UseDifferentDestinationName (QString newname)`  
Set conflict resolution to *ConflictResolution::UseDifferentDestinationName*.

`void setConflictResolve_MergeDirectories ()`  
Set conflict resolution to *ConflictResolution::MergeDirectories*.

See also `setConflictResolve_MergeDirectories_isAllowed`.

`bool setConflictResolve_MergeDirectories_checkAllowed ()`  
Checks if conflict resolution *ConflictResolution::MergeDirectories* is allowed.



## Public Members

QList<QPair<*sh::filesystem::DetailColumn\**, QVariant>> **\_detailvals**

quint64 **\_cntItems** = 0  
Number of items to be transferred in this step (used for progress monitoring).

quint64 **\_cntBytes** = 0  
Number of byte to be transferred in this step (used for progress monitoring).

bool **\_isExpanded** = false  
Bookkeeping in *FilesystemOperationTransfers*.

QList<*OperationStepClass\**> **\_expandnodes**  
Bookkeeping in *FilesystemOperationTransfers*.

*OperationStepClass* \* **\_parentnode** = 0  
Bookkeeping in *FilesystemOperationTransfers*.

bool **\_deleteonunexpand** = true  
Bookkeeping in *FilesystemOperationTransfers*.

QList<*OperationStepClass\**> **\_withExpansion**

### 10.1.171 Class *sh::filesystem::FilesystemOperationTransfers::OperationStep\_CopyDirectory*

```
class sh::filesystem::FilesystemOperationTransfers::OperationStep_CopyDirectory : public sh::fi
```

## Public Types

**enum ConflictResolution**  
Enumeration of ways how to resolve a filesystem conflict.

*Values:*

**enumerator Skip**  
Skip this element.

**enumerator OverwriteDestination**  
Overwrite the destination.

**enumerator RenameDestinationBefore**  
Rename the file at the destination before transferring.

**enumerator UseDifferentDestinationName**  
Transfer to another destination filename.

**enumerator MergeDirectories**  
Merge source into destination directory (recursively).

**enumerator Unresolved**  
No strategy.

**enumerator Indirect**  
Indirectly solved. Only set by the engine.

## Public Functions

**OperationStep\_CopyDirectory** (*FilesystemOperationTransfers* *\*transfers*,  
std::shared\_ptr<const *sh::filesystem::Eurl*> *src*,  
std::shared\_ptr<const *sh::filesystem::Eurl*> *dest*,  
QList<*OperationStepClass*\*> *withExpansion*)

void **execute** (*Operation* \**op*)  
Implement this and handle the execution part of this step here.

QList<*OperationStepClass*\*> **expand** ()  
Implement this and handle the expansion part of this step here.

int **sourcetype** ()  
Returns the node type of the source.  
  
Should be overwritten in subclasses whenever it can determine the source type faster than the infrastructure.

bool **checkconflicts** (*Operation* \**op*)  
Implement this for customizing the conflict checking.

bool **hasOwnProgressIncreaseHandling** ()  
If it takes care on its own to signal increases in the transfer progress.

void **\_unexpand** (QList<*OperationStepClass*\*> \**intlist*)  
Bookkeeping in *FilesystemOperationTransfers*.

void **\_expand** (QList<*OperationStepClass*\*> \**intlist*)  
Bookkeeping in *FilesystemOperationTransfers*.

QList<QPair<*sh::filesystem::DetailColumn*\*, QVariant>> **getFileDetails** (*sh::filesystem::Operation* \**op*,  
std::shared\_ptr<const *sh::filesystem::Eurl*> *source*)

void **applyFileDetails** (*sh::filesystem::Operation* \**op*, QList<QPair<*sh::filesystem::DetailColumn*\*,  
QVariant>> *values*, std::shared\_ptr<const *sh::filesystem::Eurl*> *destination*)

QString **conflictDescription** ()  
The description of the conflict (if any) as text.

*ConflictResolution* **conflictResolution** ()  
The currently chosen conflict resolution.

QString **conflictResolution\_renameDestinationBeforeTo** ()  
Returns new destination name (if current conflict resolution is *ConflictResolution::RenameDestinationBefore*).

QString **conflictResolution\_differentDestinationNameTo** ()  
Returns new destination name (if current conflict resolution is *ConflictResolution::UseDifferentDestinationName*).

std::shared\_ptr<const *sh::filesystem::Eurl*> **source** ()  
Returns the source location to be transferred (if specified).

std::shared\_ptr<const *sh::filesystem::Eurl*> **originalDestination** ()  
Returns the destination location (if specified).  
  
This is the complete target path, never the parent directory of the new element.  
  
This is the original destination location. See also *effectiveDestination*.

```
std::shared_ptr<const sh::filesystem::Eurl> effectiveDestination ()
    Returns the real destination location with conflict resolution applied.

void setConflictResolve_Skip ()
    Set conflict resolution to ConflictResolution::Skip.

void setConflictResolve_OverwriteDestination ()
    Set conflict resolution to ConflictResolution::OverwriteDestination.

void setConflictResolve_RenameDestinationBefore (QString newname)
    Set conflict resolution to ConflictResolution::RenameDestinationBefore.

void setConflictResolve_UseDifferentDestinationName (QString newname)
    Set conflict resolution to ConflictResolution::UseDifferentDestinationName.

void setConflictResolve_MergeDirectories ()
    Set conflict resolution to ConflictResolution::MergeDirectories.

    See also setConflictResolve_MergeDirectories_isAllowed.

bool setConflictResolve_MergeDirectories_checkAllowed ()
    Checks if conflict resolution ConflictResolution::MergeDirectories is allowed.
```

## Public Members

```
quint64 _cntItems = 0
    Number of items to be transferred in this step (used for progress monitoring).

quint64 _cntBytes = 0
    Number of byte to be transferred in this step (used for progress monitoring).

bool _isExpanded = false
    Bookkeeping in FilesystemOperationTransfers.

QList<OperationStepClass*> _expandnodes
    Bookkeeping in FilesystemOperationTransfers.

OperationStepClass * _parentnode = 0
    Bookkeeping in FilesystemOperationTransfers.

bool _deleteonunexpand = true
    Bookkeeping in FilesystemOperationTransfers.

QList<OperationStepClass*> _withExpansion
```

### 10.1.172 Class *sh::filesystem::FilesystemOperationTransfers::OperationStep\_CopyFile*

```
class sh::filesystem::FilesystemOperationTransfers::OperationStep_CopyFile : public sh::filesystem
```

## Public Types

### **enum ConflictResolution**

Enumeration of ways how to resolve a filesystem conflict.

*Values:*

#### **enumerator Skip**

Skip this element.

#### **enumerator OverwriteDestination**

Overwrite the destination.

#### **enumerator RenameDestinationBefore**

Rename the file at the destination before transferring.

#### **enumerator UseDifferentDestinationName**

Transfer to another destination filename.

#### **enumerator MergeDirectories**

Merge source into destination directory (recursively).

#### **enumerator Unresolved**

No strategy.

#### **enumerator Indirect**

Indirectly solved. Only set by the engine.

## Public Functions

**OperationStep\_CopyFile** (*FilesystemOperationTransfers* \*transfers, std::shared\_ptr<const *sh::filesystem::Eurl*> src, std::shared\_ptr<const *sh::filesystem::Eurl*> dest, QList<*OperationStepClass*\*> withExpansion)

void **execute** (*Operation* \*op)

Implement this and handle the execution part of this step here.

int **sourcetype** ()

Returns the node type of the source.

Should be overwritten in subclasses whenever it can determine the source type faster than the infrastructure.

bool **hasOwnProgressIncreaseHandling** ()

If it takes care on its own to signal increases in the transfer progress.

QList<*OperationStepClass*\*> **expand** ()

Implement this and handle the expansion part of this step here.

bool **checkconflicts** (*Operation* \*op)

Implement this for customizing the conflict checking.

void **\_unexpand** (QList<*OperationStepClass*\*> \*intlist)

Bookkeeping in *FilesystemOperationTransfers*.

void **\_expand** (QList<*OperationStepClass*\*> \*intlist)

Bookkeeping in *FilesystemOperationTransfers*.

`QList<QPair<sh::filesystem::DetailColumn*, QVariant>> getFileDetails (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> source)`

void **applyFileDetails** (*sh::filesystem::Operation* \*op, QList<QPair<*sh::filesystem::DetailColumn\**, QVariant>> values, std::shared\_ptr<const *sh::filesystem::Eurl*> destination)

QString **conflictDescription** ()  
The description of the conflict (if any) as text.

*ConflictResolution* **conflictResolution** ()  
The currently chosen conflict resolution.

QString **conflictResolution\_renameDestinationBeforeTo** ()  
Returns new destination name (if current conflict resolution is *ConflictResolution::RenameDestinationBefore*).

QString **conflictResolution\_differentDestinationNameTo** ()  
Returns new destination name (if current conflict resolution is *ConflictResolution::UseDifferentDestinationName*).

std::shared\_ptr<const *sh::filesystem::Eurl*> **source** ()  
Returns the source location to be transferred (if specified).

std::shared\_ptr<const *sh::filesystem::Eurl*> **originalDestination** ()  
Returns the destination location (if specified).  
  
This is the complete target path, never the parent directory of the new element.  
  
This is the original destination location. See also `effectiveDestination`.

std::shared\_ptr<const *sh::filesystem::Eurl*> **effectiveDestination** ()  
Returns the real destination location with conflict resolution applied.

void **setConflictResolve\_Skip** ()  
Set conflict resolution to *ConflictResolution::Skip*.

void **setConflictResolve\_OverwriteDestination** ()  
Set conflict resolution to *ConflictResolution::OverwriteDestination*.

void **setConflictResolve\_RenameDestinationBefore** (QString newname)  
Set conflict resolution to *ConflictResolution::RenameDestinationBefore*.

void **setConflictResolve\_UseDifferentDestinationName** (QString newname)  
Set conflict resolution to *ConflictResolution::UseDifferentDestinationName*.

void **setConflictResolve\_MergeDirectories** ()  
Set conflict resolution to *ConflictResolution::MergeDirectories*.  
  
See also `setConflictResolve_MergeDirectories_isAllowed`.

bool **setConflictResolve\_MergeDirectories\_checkAllowed** ()  
Checks if conflict resolution *ConflictResolution::MergeDirectories* is allowed.

## Public Members

quint64 **\_cntItems** = 0  
Number of items to be transferred in this step (used for progress monitoring).

quint64 **\_cntBytes** = 0  
Number of byte to be transferred in this step (used for progress monitoring).

bool **\_isExpanded** = false  
Bookkeeping in *FilesystemOperationTransfers*.

QList<*OperationStepClass*\*> **\_expandnodes**  
Bookkeeping in *FilesystemOperationTransfers*.

*OperationStepClass* \* **\_parentnode** = 0  
Bookkeeping in *FilesystemOperationTransfers*.

bool **\_deleteonunexpand** = true  
Bookkeeping in *FilesystemOperationTransfers*.

QList<*OperationStepClass*\*> **\_withExpansion**

### 10.1.173 Class sh::filesystem::FilesystemOperationTransfers::OperationStep\_CopyLink

```
class sh::filesystem::FilesystemOperationTransfers::OperationStep_CopyLink : public sh::filesystem
```

## Public Types

enum **ConflictResolution**  
Enumeration of ways how to resolve a filesystem conflict.

*Values:*

**enumerator Skip**  
Skip this element.

**enumerator OverwriteDestination**  
Overwrite the destination.

**enumerator RenameDestinationBefore**  
Rename the file at the destination before transferring.

**enumerator UseDifferentDestinationName**  
Transfer to another destination filename.

**enumerator MergeDirectories**  
Merge source into destination directory (recursively).

**enumerator Unresolved**  
No strategy.

**enumerator Indirect**  
Indirectly solved. Only set by the engine.

## Public Functions

**OperationStep\_CopyLink** (*FilesystemOperationTransfers* \*transfers, std::shared\_ptr<const sh::filesystem::Eurl> src, std::shared\_ptr<const sh::filesystem::Eurl> dest, QList<OperationStepClass\*> withExpansion)

void **execute** (*Operation* \*op)

Implement this and handle the execution part of this step here.

int **sourcetype** ()

Returns the node type of the source.

Should be overwritten in subclasses whenever it can determine the source type faster than the infrastructure.

QList<OperationStepClass\*> **expand** ()

Implement this and handle the expansion part of this step here.

bool **checkconflicts** (*Operation* \*op)

Implement this for customizing the conflict checking.

bool **hasOwnProgressIncreaseHandling** ()

If it takes care on its own to signal increases in the transfer progress.

void **\_unexpand** (QList<OperationStepClass\*> \*intlist)

Bookkeeping in *FilesystemOperationTransfers*.

void **\_expand** (QList<OperationStepClass\*> \*intlist)

Bookkeeping in *FilesystemOperationTransfers*.

QList<QPair<sh::filesystem::DetailColumn\*, QVariant>> **getFileDetails** (sh::filesystem::Operation \*op, std::shared\_ptr<const sh::filesystem::Eurl> source)

void **applyFileDetails** (sh::filesystem::Operation \*op, QList<QPair<sh::filesystem::DetailColumn\*, QVariant>> values, std::shared\_ptr<const sh::filesystem::Eurl> destination)

QString **conflictDescription** ()

The description of the conflict (if any) as text.

*ConflictResolution* **conflictResolution** ()

The currently chosen conflict resolution.

QString **conflictResolution\_renameDestinationBeforeTo** ()

Returns new destination name (if current conflict resolution is *ConflictResolution::RenameDestinationBefore*).

QString **conflictResolution\_differentDestinationNameTo** ()

Returns new destination name (if current conflict resolution is *ConflictResolution::UseDifferentDestinationName*).

std::shared\_ptr<const sh::filesystem::Eurl> **source** ()

Returns the source location to be transferred (if specified).

std::shared\_ptr<const sh::filesystem::Eurl> **originalDestination** ()

Returns the destination location (if specified).

This is the complete target path, never the parent directory of the new element.

This is the original destination location. See also effectiveDestination.

`std::shared_ptr<const sh::filesystem::Eurl> effectiveDestination ()`  
Returns the real destination location with conflict resolution applied.

`void setConflictResolve_Skip ()`  
Set conflict resolution to *ConflictResolution::Skip*.

`void setConflictResolve_OverwriteDestination ()`  
Set conflict resolution to *ConflictResolution::OverwriteDestination*.

`void setConflictResolve_RenameDestinationBefore (QString newname)`  
Set conflict resolution to *ConflictResolution::RenameDestinationBefore*.

`void setConflictResolve_UseDifferentDestinationName (QString newname)`  
Set conflict resolution to *ConflictResolution::UseDifferentDestinationName*.

`void setConflictResolve_MergeDirectories ()`  
Set conflict resolution to *ConflictResolution::MergeDirectories*.  
  
See also `setConflictResolve_MergeDirectories_isAllowed`.

`bool setConflictResolve_MergeDirectories_checkAllowed ()`  
Checks if conflict resolution *ConflictResolution::MergeDirectories* is allowed.

## Public Members

`quint64 _cntItems = 0`  
Number of items to be transferred in this step (used for progress monitoring).

`quint64 _cntBytes = 0`  
Number of byte to be transferred in this step (used for progress monitoring).

`bool _isExpanded = false`  
Bookkeeping in *FilesystemOperationTransfers*.

`QList<OperationStepClass*> _expandnodes`  
Bookkeeping in *FilesystemOperationTransfers*.

`OperationStepClass * _parentnode = 0`  
Bookkeeping in *FilesystemOperationTransfers*.

`bool _deleteonunexpand = true`  
Bookkeeping in *FilesystemOperationTransfers*.

`QList<OperationStepClass*> _withExpansion`

### 10.1.174 Class *sh::filesystem::FilesystemOperationTransfers::OperationStep\_DeleteItem*

```
class sh::filesystem::FilesystemOperationTransfers::OperationStep_DeleteItem: public sh::file
```



## Public Types

### enum ConflictResolution

Enumeration of ways how to resolve a filesystem conflict.

*Values:*

#### enumerator Skip

Skip this element.

#### enumerator OverwriteDestination

Overwrite the destination.

#### enumerator RenameDestinationBefore

Rename the file at the destination before transferring.

#### enumerator UseDifferentDestinationName

Transfer to another destination filename.

#### enumerator MergeDirectories

Merge source into destination directory (recursively).

#### enumerator Unresolved

No strategy.

#### enumerator Indirect

Indirectly solved. Only set by the engine.

## Public Functions

**OperationStep\_DeleteItem** (*FilesystemOperationTransfers* \*transfers, std::shared\_ptr<const sh::filesystem::Eurl> src, std::shared\_ptr<const sh::filesystem::Eurl> dest, QList<OperationStepClass\*> withExpansion)

void **execute** (*Operation* \*op)

Implement this and handle the execution part of this step here.

bool **checkconflicts** (*Operation* \*op)

Implement this for customizing the conflict checking.

int **sourcetype** ()

Returns the node type of the source.

Should be overwritten in subclasses whenever it can determine the source type faster than the infrastructure.

QList<OperationStepClass\*> **expand** ()

Implement this and handle the expansion part of this step here.

bool **hasOwnProgressIncreaseHandling** ()

If it takes care on its own to signal increases in the transfer progress.

void **\_unexpand** (QList<OperationStepClass\*> \*intlist)

Bookkeeping in *FilesystemOperationTransfers*.

void **\_expand** (QList<OperationStepClass\*> \*intlist)

Bookkeeping in *FilesystemOperationTransfers*.

`QList<QPair<sh::filesystem::DetailColumn*, QVariant>> getFileDetails (sh::filesystem::Operation *op,  
std::shared_ptr<const sh::filesystem::Eurl>  
source)`

`void applyFileDetails (sh::filesystem::Operation *op, QList<QPair<sh::filesystem::DetailColumn*,  
QVariant>> values, std::shared_ptr<const sh::filesystem::Eurl> destina-  
tion)`

`QString conflictDescription ()`  
The description of the conflict (if any) as text.

`ConflictResolution conflictResolution ()`  
The currently chosen conflict resolution.

`QString conflictResolution_renameDestinationBeforeTo ()`  
Returns new destination name (if current conflict resolution is *ConflictResolu-  
tion::RenameDestinationBefore*).

`QString conflictResolution_differentDestinationNameTo ()`  
Returns new destination name (if current conflict resolution is *ConflictResolu-  
tion::UseDifferentDestinationName*).

`std::shared_ptr<const sh::filesystem::Eurl> source ()`  
Returns the source location to be transferred (if specified).

`std::shared_ptr<const sh::filesystem::Eurl> originalDestination ()`  
Returns the destination location (if specified).

This is the complete target path, never the parent directory of the new element.

This is the original destination location. See also `effectiveDestination`.

`std::shared_ptr<const sh::filesystem::Eurl> effectiveDestination ()`  
Returns the real destination location with conflict resolution applied.

`void setConflictResolve_Skip ()`  
Set conflict resolution to *ConflictResolution::Skip*.

`void setConflictResolve_OverwriteDestination ()`  
Set conflict resolution to *ConflictResolution::OverwriteDestination*.

`void setConflictResolve_RenameDestinationBefore (QString newname)`  
Set conflict resolution to *ConflictResolution::RenameDestinationBefore*.

`void setConflictResolve_UseDifferentDestinationName (QString newname)`  
Set conflict resolution to *ConflictResolution::UseDifferentDestinationName*.

`void setConflictResolve_MergeDirectories ()`  
Set conflict resolution to *ConflictResolution::MergeDirectories*.

See also `setConflictResolve_MergeDirectories_isAllowed`.

`bool setConflictResolve_MergeDirectories_checkAllowed ()`  
Checks if conflict resolution *ConflictResolution::MergeDirectories* is allowed.

## Public Members

quint64 **\_cntItems** = 0  
 Number of items to be transferred in this step (used for progress monitoring).

quint64 **\_cntBytes** = 0  
 Number of byte to be transferred in this step (used for progress monitoring).

bool **\_isExpanded** = false  
 Bookkeeping in *FilesystemOperationTransfers*.

QList<*OperationStepClass*\*> **\_expandnodes**  
 Bookkeeping in *FilesystemOperationTransfers*.

*OperationStepClass* \* **\_parentnode** = 0  
 Bookkeeping in *FilesystemOperationTransfers*.

bool **\_deleteonunexpand** = true  
 Bookkeeping in *FilesystemOperationTransfers*.

QList<*OperationStepClass*\*> **\_withExpansion**

### 10.1.175 Class sh::filesystem::FilesystemOperationTransfers::OperationStep\_RenameItem

```
class sh::filesystem::FilesystemOperationTransfers::OperationStep_RenameItem: public sh::filesystem::OperationStep
```

## Public Types

enum **ConflictResolution**  
 Enumeration of ways how to resolve a filesystem conflict.

*Values:*

enumerator **Skip**  
 Skip this element.

enumerator **OverwriteDestination**  
 Overwrite the destination.

enumerator **RenameDestinationBefore**  
 Rename the file at the destination before transferring.

enumerator **UseDifferentDestinationName**  
 Transfer to another destination filename.

enumerator **MergeDirectories**  
 Merge source into destination directory (recursively).

enumerator **Unresolved**  
 No strategy.

enumerator **Indirect**  
 Indirectly solved. Only set by the engine.

## Public Functions

**OperationStep\_RenameItem** (*FilesystemOperationTransfers* \*transfers, std::shared\_ptr<const sh::filesystem::Eurl> src, QString destpath, QList<OperationStepClass\*> withExpansion)

QList<OperationStepClass\*> **expand** ()  
Implement this and handle the expansion part of this step here.

void **execute** (*Operation* \*op)  
Implement this and handle the execution part of this step here.

bool **checkconflicts** (*Operation* \*op)  
Implement this for customizing the conflict checking.

bool **hasOwnProgressIncreaseHandling** ()  
If it takes care on its own to signal increases in the transfer progress.

void **\_unexpand** (QList<OperationStepClass\*> \*intlist)  
Bookkeeping in *FilesystemOperationTransfers*.

void **\_expand** (QList<OperationStepClass\*> \*intlist)  
Bookkeeping in *FilesystemOperationTransfers*.

QList<QPair<sh::filesystem::DetailColumn\*, QVariant>> **getFileDetails** (sh::filesystem::Operation \*op, std::shared\_ptr<const sh::filesystem::Eurl> source)

void **applyFileDetails** (sh::filesystem::Operation \*op, QList<QPair<sh::filesystem::DetailColumn\*, QVariant>> values, std::shared\_ptr<const sh::filesystem::Eurl> destination)

QString **conflictDescription** ()  
The description of the conflict (if any) as text.

*ConflictResolution* **conflictResolution** ()  
The currently chosen conflict resolution.

QString **conflictResolution\_renameDestinationBeforeTo** ()  
Returns new destination name (if current conflict resolution is *ConflictResolution::RenameDestinationBefore*).

QString **conflictResolution\_differentDestinationNameTo** ()  
Returns new destination name (if current conflict resolution is *ConflictResolution::UseDifferentDestinationName*).

std::shared\_ptr<const sh::filesystem::Eurl> **source** ()  
Returns the source location to be transferred (if specified).

std::shared\_ptr<const sh::filesystem::Eurl> **originalDestination** ()  
Returns the destination location (if specified).  
  
This is the complete target path, never the parent directory of the new element.  
  
This is the original destination location. See also effectiveDestination.

std::shared\_ptr<const sh::filesystem::Eurl> **effectiveDestination** ()  
Returns the real destination location with conflict resolution applied.

int **sourcetype** ()  
Returns the node type of the source.

Should be overwritten in subclasses whenever it can determine the source type faster than the infrastructure.

```
void setConflictResolve_Skip ()
    Set conflict resolution to ConflictResolution::Skip.

void setConflictResolve_OverwriteDestination ()
    Set conflict resolution to ConflictResolution::OverwriteDestination.

void setConflictResolve_RenameDestinationBefore (QString newname)
    Set conflict resolution to ConflictResolution::RenameDestinationBefore.

void setConflictResolve_UseDifferentDestinationName (QString newname)
    Set conflict resolution to ConflictResolution::UseDifferentDestinationName.

void setConflictResolve_MergeDirectories ()
    Set conflict resolution to ConflictResolution::MergeDirectories.

    See also setConflictResolve_MergeDirectories_isAllowed.

bool setConflictResolve_MergeDirectories_checkAllowed ()
    Checks if conflict resolution ConflictResolution::MergeDirectories is allowed.
```

## Public Members

```
quint64 _cntItems = 0
    Number of items to be transferred in this step (used for progress monitoring).

quint64 _cntBytes = 0
    Number of byte to be transferred in this step (used for progress monitoring).

bool _isExpanded = false
    Bookkeeping in FilesystemOperationTransfers.

QList<OperationStepClass*> _expandnodes
    Bookkeeping in FilesystemOperationTransfers.

OperationStepClass * _parentnode = 0
    Bookkeeping in FilesystemOperationTransfers.

bool _deleteonunexpand = true
    Bookkeeping in FilesystemOperationTransfers.

QList<OperationStepClass*> _withExpansion
```

### 10.1.176 Class sh::filesystem::FilesystemOperationTransfers::SingleStepMonitor

```
class sh::filesystem::FilesystemOperationTransfers::SingleStepMonitor : public sh::filesystem::FilesystemOperationTransfers
    Used for managing detailed progress changes about a single step.
```

### Public Functions

```
SingleStepMonitor (OperationStepClass *step)
void respectCancel ()
void incrementTransferredBytes (qint64 donebytes)
~SingleStepMonitor ()
```

### Private Members

```
OperationStepClass *step
qint64 _donebytes = 0
```

## 10.1.177 Class sh::filesystem::FilesystemOperation::MyHandlerTransfer

```
class sh::filesystem::FilesystemOperation::MyHandlerTransfer : public sh::filesystem::FilesystemHandler
```

### Public Functions

```
MyHandlerTransfer (sh::filesystem::FilesystemOperationProgressMonitor *progressmon = 0)
~MyHandlerTransfer ()
void respectCancel ()
void incrementTransferredBytes (qint64 donebytes)
```

### Private Members

```
sh::filesystem::FilesystemOperationProgressMonitor *_progressmon
```

## 10.1.178 Class sh::filesystem::LoadOnDemandPlaceholderFilesystemNode

```
class sh::filesystem::LoadOnDemandPlaceholderFilesystemNode : public sh::filesystem::FilesystemNode
```

A special node, which is unselectable and shows a 'loading' label.

It does not correspond to a file, directory or similar which actually exists. It is also never parent or a child of another node.

They are used internally by the filesystem model.

### Public Functions

```
QString displayName ()
    Returns the display name (what the user interface displays).
void requestDetails (bool force)
    Requests to fetch details for this node.
LoadOnDemandPlaceholderFilesystemNode (std::shared_ptr<sh::filesystem::FilesystemNode>
    parent)
    Constructed only by the infrastructure and made available otherwise.
```

`std::shared_ptr<const sh::filesystem::Eurl> eurl ()`  
 Returns the *sh::filesystem::Eurl* entry address.

*sh::filesystem::FilesystemModel* \***model** ()  
 Convenience shortcut to the *sh::filesystem::FilesystemModel*.

void **setDisplayname** (QString *displayname*)  
 Sets the display name (what the user interface displays).

*sh::filesystem::FilesystemHandler* \***handler** ()  
 Returns the handler which is responsible for this node. This should be the same as *sh::filesystem::FilesystemHandlerRegister.findHandler(eurl.scheme)*.

int **nodetype** ()  
 Returns the FilesystemNodeType node type.

int **childnodeCount** ()  
 Returns the number of children nodes.

`std::shared_ptr<sh::filesystem::FilesystemNode> childnode (int i)`  
 Returns i-th child node.

*sh::filesystem::ModelBackedFilesystemNodeList* \***childnodes** ()  
 Returns the list of children.

`std::shared_ptr<sh::filesystem::FilesystemNode> parentnode () const`  
 Returns the parent node.

This is 0 if the node does not exist in the model (i.e. not yet inserted or removed afterwards). In most cases, this can be interpreted as ‘file currently does not exist’.

int **index** ()  
 Returns the index of this node in the parent’s list of children.

void **addChild** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> *node*)  
 Adds a new child. Not allowed to call more than once for a node.

void **removeChild** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> *node*)  
 Removes a new child. Not allowed to call more than once for a node.

bool **isFetchingSubitems** ()  
 Checks if currently subitems are fetched (i.e. the ‘loading’ node is shown).

QIcon **icon** ()  
 Returns the node icon.

void **setIcon** (QIcon *icon*)  
 Sets the node icon.

bool **isHidden** ()  
 Returns if the node is hidden.

Note: It’s not difficult for the user to also show the hidden items.

void **setHidden** (bool *v*)  
 Sets if the node is hidden.

See also *isHidden()*.

void **addDetail** (std::shared\_ptr<*sh::filesystem::DetailColumn*> *column*)  
 Adds a detail to this node.

`std::shared_ptr<sh::filesystem::DetailColumn> getDetailColumnByIndex (int index)`  
 Returns the i-th detail column registered to this node.

int **getDetailColumnsCount** () **const**

Returns the number of detail columns registered to this node.

bool **isRootNode** () **const**

Checks if this is the root node of the model.

bool **isConfigured** () **const**

Checks if this node was already configured by a handler.

bool **isAlive** () **const**

Checks if this node currently exists in the model.

QList<std::shared\_ptr<sh::filesystem::FileSystemNode>> **linkDestinationNodes** () **const**

Returns the list of link source nodes. If this node is a link, the link destination nodes may influence the behavior and appearance of this node.

This information does not come from inside but must be set from outside the model implementation.

void **setLinkDestinationNodes** (QList<std::shared\_ptr<sh::filesystem::FileSystemNode>>  
linkdestinations)

Sets the list of link destination nodes. If this node is a link, the link destination nodes may influence the behavior and appearance of this node.

void **requestContainedItems** (sh::filesystem::FileSystemNodeType type = FileSystemNode-  
Type::NONE)

Requests to fetch the children of this node with help of the filesystem handler.

## Signals

void **removed** ()

Is emitted when this node is not placed in the tree anymore. This does not mean the physical deletion of the instance, but just means it is not alive from now on.

void **\_detailsAvailable** ()

Is emitted when values for detail columns arrived.

void **iconChanged** ()

Is emitted when the icon changed.

void **isHiddenChanged** ()

Is emitted when the hidden flag changed.

## Private Members

bool **loading**

## Friends

**friend class** sh::filesystem::FileSystemModel



### 10.1.179 Class `sh::filesystem::ModelBackedFilesystemNodeList`

**class** `sh::filesystem::ModelBackedFilesystemNodeList` : public `sh::filesystem::FilesystemNodeList`  
 This *FilesystemNodeList* subclass builds a part of the filesystem model.

The listed nodes are actually owned and handled by a `sh::filesystem::FilesystemNode`. Node additions and removals will directly influence the model.

#### Public Functions

**ModelBackedFilesystemNodeList** (`sh::filesystem::FilesystemNode *node`)

Constructed only internally.

void **addItem** (QSet<std::shared\_ptr<*FilesystemNode*>> *nodes*)

void **addItem** (std::shared\_ptr<*FilesystemNode*> *node*)

void **removeItems** (QSet<std::shared\_ptr<*FilesystemNode*>> *nodes*)

void **removeItem** (std::shared\_ptr<*FilesystemNode*> *node*)

void **resetItems** (*sh::filesystem::FilesystemNodeType* *type*, QSet<std::shared\_ptr<*FilesystemNode*>> *nodes*)

std::shared\_ptr<*FilesystemNode*> **mynode** ()

Returns the node which owns this children list. The result may be 0 for non model-backed lists.

void **clear** ()

Clears the entire list. This is not a high-level operation, but something only used internally (not part of the interface).

void **addPermanentItem** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> *node*)

Adds an item to this list. In a model-backed list, this triggers the internal model mounting calls. It is not allowed to call this method twice with the same node.

This list is safe against removing from later item lists. Only `removePermanentItem` removes it. See also `addItem`.

void **removePermanentItem** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> *node*)

Removes a permanent node from this list. In a model-backed list, this triggers the internal model unmounting calls. It is not allowed to call this method with a node, which is not contained in that list.

void **addItem** (QSet<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> *nodes*) = 0

Adds items to this list. In a model-backed list, this triggers the internal model mounting calls. It is not allowed to call this method twice with the same node.

void **addItem** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> *node*) = 0

Adds an item to this list. In a model-backed list, this triggers the internal model mounting calls. It is not allowed to call this method twice with the same node.

void **removeItems** (QSet<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> *nodes*) = 0

Removes nodes from this list. In a model-backed list, this triggers the internal model unmounting calls. It is not allowed to call this method with a node, which is not contained in that list.

void **removeItem** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> *node*) = 0

Removes a node from this list. In a model-backed list, this triggers the internal model unmounting calls. It is not allowed to call this method with a node, which is not contained in that list.

void **resetItems** (*sh::filesystem::FilesystemNodeType* *type*, QSet<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> *nodes*) = 0

Resets the list for a given node type to a given new list of children nodes.

bool **contains** (std::shared\_ptr<*FilesystemNode*> *node*)

Checks if this list contains a given node.

const QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> **\*nodes** ()

Returns the list of nodes currently stored in this instance.

### Private Functions

void **addItem\_helper** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> *node*)

### Private Members

QSet<*sh::filesystem::FilesystemNode*\*> **\_permanentnodes**

*sh::filesystem::FilesystemNode* **\*\_mynode**

*sh::filesystem::FilesystemModel* **\*mymodel**

## 10.1.180 Class *sh::filesystem::Operation*

**class** *sh::filesystem::Operation* : **public** QObject

A operation surrounds a completed series of steps in the filesystem.

For some steps to execute in the filesystem (i.e. not only the local one but the entire *Eurl* universe), it fetches intermediate files, caches them and writes them back to their real location afterwards.

It provides transaction-like operations for committing/dropping those intermediate files. A high-level interface for file management is available in *filesystem*.

It is allowed to create new instances from scratch, but you should check if you implicitly have an instance available in your situation, or if *sh::tools::OperationsCache* is an option.

### Public Functions

**Operation** (QObject *\*parent* = 0)

Often constructed by the infrastructure and made available, but can also be constructed directly.

quint64 **getFreeCacheSpace** ()

Returns the free disk space (in bytes) available for operations like *fetchContainerFile()* or *fetchFile()*.

QString **fetchContainerFile** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*, QString *name-hint*)

Fetches the container file for a *eurl* locally and returns the local path. Use this with care and only if needed. In many cases working with streams is the better way!

QString **fetchContainerFile** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*)

QString **fetchFile** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*, QString *namehint*)

Fetches a file locally and returns the local path. Use this with care and only if needed. In many cases working with streams is the better way!

QString **fetchFile** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*)

void **abort** ()

Aborts transaction dropping all pending changes.

void **commit** ()

Commits transaction applying all pending changes.

```

void enableDetailsCache ()
    Enable details caching.

bool isDetailsCacheEnabled ()
    Is details caching enabled?

void storeDetailInCache (std::shared_ptr<const sh::filesystem::Eurl> eurl,
                        sh::filesystem::DetailColumn *column, QVariant value)
    Stores a column detail in cache.

QVariant getDetailFromCache (std::shared_ptr<const sh::filesystem::Eurl> eurl, const
                             sh::filesystem::DetailColumn *column)
    Gets a column value from cache.

QList<std::shared_ptr<const sh::filesystem::Eurl>> pendingCommits ()
    Returns a list of sh::filesystem::Eurl items which are marked for commit.

QString getTempDir ()
    Creates a fresh temporary folder and returns the path to it.

void setCustomData (std::shared_ptr<const sh::filesystem::Eurl> eurl, QString key, QVariant data)
    Stores custom data.

QVariant getCustomData (std::shared_ptr<const sh::filesystem::Eurl> eurl, QString key)
    Gets stored custom data.

void setMaxAllowedSizeRatioPerPart (double v)
    Sets a maximum part of available disk space for usage. This is exotic functionality you typically don't
    need. .

```

#### Parameters

- v: A ratio (between 0 and 1) of the available disk space which one part maximally may cost.

```

sh::filesystem::FilesystemOperation *filesystem ()
    Gets the sh::filesystem::FilesystemOperation filesystem operation object.

```

```

~Operation ()

```

#### Public Static Functions

```

void writeIODeviceToFile (QIODevice *content, QString filepath)
    Low-level function for writing a QIODevice content to a local path.

```

#### Private Members

```

QHash<std::shared_ptr<const sh::filesystem::Eurl>, QString> files
QSet<std::shared_ptr<const sh::filesystem::Eurl>> fileIsTemporary
QList<QString> tempDirs
bool _detailsCacheEnabled = false
QHash<std::shared_ptr<const sh::filesystem::Eurl>, QMap<QString, QVariant>> _customData
QHash<const sh::filesystem::DetailColumn*, QHash<std::shared_ptr<const sh::filesystem::Eurl>, QVariant>> _detailsCa
sh::filesystem::FilesystemOperation *_filesystemOperation
double _maxAllowedSizeRatioPerPart

```

QMutex **operationlock**

### Private Static Attributes

QMutex **\_tempfilemutex**

QString **\_tempdir**

**class MaxAllowedSizeRatioPerPartExceededException** : public *sh::exceptions::IOException*  
IO exception raised when the value in *setMaxAllowedSizeRatioPerPart()* was exceeded.

### Public Functions

**MaxAllowedSizeRatioPerPartExceededException** ()

QString **message** () const

QString **name** () const

QString **classes** () const

QString **details** () const

QString **callstack** () const

QString **auxiliary** () const

QString **customValue** (QString *key*) const

bool **isRuntimeException** () const

bool **isProgramException** () const

bool **isRetryable** () const

bool **isResumeable** () const

bool **isDetailsAreInteresting** () const

int **autoRetryRecommendedIn** () const

void **setResumeable** (bool *v*)

void **setReTryable** (bool *v*)

void **setCustomValue** (QString *key*, QString *value*)

bool **isClass** (QString *classname*)

*sh::exceptions::ExceptionData* **data** ()

### Public Static Functions

template<class **Handler**>  
std::shared\_ptr<RegisterHandler<*Handler*>> **createRegisterHandler** (*Handler* *handler*,  
QStack<*Handler*>  
\**stack*)

void **executeGuarded** (std::function<void>  
> *fct* int *flags* = 0) Executes some code with some standard exection handling around.

#### Parameters

- *fct*: The code to execute guarded.

- `flags`: Flags of *ExecuteGuardFlag* for choosing the behavior.

```
void executeGuarded_errorpanel (std::function<void>
    > fcn, int flags = 0)
```

```
QString __demangle (QString l)
```

```
void exceptionDialog (sh::exceptions::Exception &e, bool *doRetry)
```

### Public Static Attributes

```
const QString UNDEFINED_ERROR_OCCURRED = QObject::tr("An unspecified error occurred.")
```

```
const QString SHALLOT_MUST_CLOSE_TEXT = QObject::tr("Your Shallot process is disturbed by an error and needs to be restarted.")
```

```
const QString Value_isShallotException = "_isShallotException"
```

## 10.1.181 Class *sh::filesystem::Operation::MaxAllowedSizeRatioPerPartExceededException*

**class** *sh::filesystem::Operation::MaxAllowedSizeRatioPerPartExceededException* : public *sh::exceptions::Exception*  
 IO exception raised when the value in *setMaxAllowedSizeRatioPerPart()* was exceeded.

### Public Functions

```
MaxAllowedSizeRatioPerPartExceededException ()
```

```
QString message () const
```

```
QString name () const
```

```
QString classes () const
```

```
QString details () const
```

```
QString callstack () const
```

```
QString auxiliary () const
```

```
QString customValue (QString key) const
```

```
bool isRuntimeException () const
```

```
bool isProgramException () const
```

```
bool isRetryable () const
```

```
bool isResumeable () const
```

```
bool isDetailsAreInteresting () const
```

```
int autoRetryRecommendedIn () const
```

```
void setResumeable (bool v)
```

```
void setReTryable (bool v)
```

```
void setCustomValue (QString key, QString value)
```

```
bool isClass (QString classname)
```

```
sh::exceptions::ExceptionData data ()
```

## Public Static Functions

```
template<class Handler>
std::shared_ptr<RegisterHandler<Handler>> createRegisterHandler (Handler handler, QStack<Handler> *stack)
```

```
void executeGuarded (std::function<void>
    > fct int flags = 0 Executes some code with some standard execution handling around.
```

### Parameters

- *fct*: The code to execute guarded.
- *flags*: Flags of *ExecuteGuardFlag* for choosing the behavior.

```
void executeGuarded_errorpanel (std::function<void>
    > fct int flags = 0
```

```
QString _demangle (QString l)
```

```
void exceptionDialog (sh::exceptions::Exception &e, bool *doRetry)
```

## Public Static Attributes

```
const QString UNDEFINED_ERROR_OCCURRED = QObject::tr("An unspecified error occurred.")
```

```
const QString SHALLOT_MUST_CLOSE_TEXT = QObject::tr("Your Shallot process is disturbed by an error and needs to be closed.")
```

```
const QString Value_isShallotException = "_isShallotException"
```

### 10.1.182 Class *sh::filesystemhandlers::ActionMountGnomeIOLocation*

```
class sh::filesystemhandlers::ActionMountGnomeIOLocation : public sh::actions::ActionActionItem
    Action for mounting a GIO location.
```

### Public Functions

```
ActionMountGnomeIOLocation (QList<std::shared_ptr<sh::filesystem::FilesystemNode>> nodes)
    Constructed only internally.
```

```
void execute ()
    Executes this action.
```

```
void execute (sh::actions::ActionExecutionInfo *info)
    Executes this action.
```

```
QKeySequence shortcuthint ()
    Returns the keyboard shortcut for triggering this action.
```

```
bool shortcuthintTriggersOnCurrentDirectory ()
    Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).
```

```
void setShortcuthint (QKeySequence shortcut, bool triggersOnCurrentDirectory = false)
    Sets the keyboard shortcut for triggering this action.
```

```
QString text ()
    Returns the displayed text for this action.
```

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<AbstractActionItem> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<AbstractActionItem> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

## Public Static Functions

void **offerMountGVolumeUuid** (std::shared\_ptr<sh::filesystem::FilesystemNode> node, QString gvolumeuuid)  
void **offerMountLocation** (std::shared\_ptr<sh::filesystem::FilesystemNode> node, QString location)  
void **unofferMount** (std::shared\_ptr<sh::filesystem::FilesystemNode> node)  
void **doInitialize** ()  
void **doShutdown** ()  
QString **mount** (sh::actions::ActionExecutionInfo \*info, QString gvolumeuuid, QString location, std::shared\_ptr<filesystem::FilesystemNode> node)

### 10.1.183 Class sh::filesystemhandlers::ActionMountNetworkGnomeIOLocation

**class** sh::filesystemhandlers::ActionMountNetworkGnomeIOLocation : public sh::actions::ActionActionItem  
Action for mounting a GIO network location.

## Public Functions

**ActionMountNetworkGnomeIOLocation** ()  
Constructed only internally.  
void **execute** ()  
Executes this action.  
void **execute** (sh::actions::ActionExecutionInfo \*info)  
Executes this action.  
QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.  
bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).  
void **setShortcuthint** (QKeySequence shortcut, bool triggersOnCurrentDirectory = false)  
Sets the keyboard shortcut for triggering this action.  
QString **text** ()  
Returns the displayed text for this action.  
QString **icon** ()  
Returns the icon for this action.  
bool **enabled** ()  
Checks if this action is enabled.  
bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).



bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<AbstractActionItem> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<AbstractActionItem> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

### 10.1.184 Class `sh::filesystemhandlers::ActionUnmountGnomeIOLocation`

**class** `sh::filesystemhandlers::ActionUnmountGnomeIOLocation` : **public** `sh::actions::ActionActionItem`  
Action for unmounting a GIO location.

#### Public Functions

**ActionUnmountGnomeIOLocation** (`QList<std::shared_ptr<sh::filesystem::FilesystemNode>>`  
*nodes*)

Constructed only internally.

**void execute** ()

Executes this action.

**void execute** (`sh::actions::ActionExecutionInfo *info`)

Executes this action.

**QKeySequence shortcutHint** ()

Returns the keyboard shortcut for triggering this action.

**bool shortcutHintTriggersOnCurrentDirectory** ()

Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

**void setShortcutHint** (`QKeySequence shortcut`, `bool triggersOnCurrentDirectory = false`)

Sets the keyboard shortcut for triggering this action.

**QString text** ()

Returns the displayed text for this action.

**QString icon** ()

Returns the icon for this action.

**bool enabled** ()

Checks if this action is enabled.

**bool isChecked** ()

Checks if this action is checked (has a cross in the ui).

**bool isCheckable** ()

Checks if this action is checkable (can have a cross in the ui).

**int defaultActionPrecedence** () **const**

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

**void setText** (`QString text`)

Sets the displayed text.

**void setIcon** (`QString icon`)

Sets the icon.

**void setEnabled** (`bool enabled`)

Sets if the item is enabled.

**void setChecked** (`bool checked`)

Sets if the item is checked (has a cross in the ui).

```

void setVisible (bool visible)
    Sets the visibility of this item.

bool visible ()
    Checks the visibility of this item (non-recursively).

std::weak_ptr<AbstractActionItem> parentAction ()
    Returns the parent action, if it is added to a container.

void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.

```

## Signals

```

void changed ()
    Emits when some data changed.

```

## Public Static Functions

```

void offerUnmountLocation (std::shared_ptr<sh::filesystem::FilesystemNode> node, QString location, bool staticmode = false)

void unofferUnmount (std::shared_ptr<sh::filesystem::FilesystemNode> node)

void doInitialize ()

void doShutdown ()

```

### 10.1.185 Class *sh::filesystemhandlers::ArchiveFilesystemHandler*

```

class sh::filesystemhandlers::ArchiveFilesystemHandler : public sh::filesystem::FilesystemHandler
    A filesystem handler for archive files of some formats.

    This implementation uses process calls to some external utility tools.

```

## Public Functions

```
ArchiveFilesystemHandler (sh::filesystem::FilesystemModel *model)

void itemlist (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
              sh::filesystem::FilesystemNodeType type, sh::filesystem::FilesystemNodeListEditor
              *list)

sh::filesystem::FilesystemNodeType getType (sh::filesystem::Operation *op, std::shared_ptr<const
                                             sh::filesystem::Eurl> eurl)

qint64 getSize (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl)

QDateTime getMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl)

void setMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
              QDateTime time)

QString getMimeType (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                     eurl)

QString getLinkTarget (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                       eurl)

bool canGetFileContent (sh::filesystem::Operation *op, std::shared_ptr<const
                       sh::filesystem::Eurl> eurl)

std::shared_ptr<QIODevice> getFileContent (sh::filesystem::Operation *op,
                                           std::shared_ptr<const sh::filesystem::Eurl> eurl)

bool canCreateDirectory (sh::filesystem::Operation *op, std::shared_ptr<const
                          sh::filesystem::Eurl> eurl)

void createDirectory (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                      eurl)

bool canCreateLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                     eurl)

void createLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
                 QString target)

bool canCreateFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                     eurl)

void createFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
                 QIODevice *content, HandlerTransfer *handlertransfer)

bool canDeleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                     eurl)

void deleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl)

bool canRenameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                     src)

void renameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> src,
                 QString destpath)

QList<std::shared_ptr<sh::actions::AbstractActionItem>> getActions (const
                                                                    QList<std::shared_ptr<const
                                                                    sh::filesystem::Eurl>> eurls)

void itemlist (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
              sh::filesystem::FilesystemNodeType type, sh::filesystem::FilesystemNodeListEditor
              *list) = 0
```

Determine a list of subelements in a certain directory.

void **configureItems** (*sh::filesystem::Operation* \*op, QList<std::shared\_ptr<FilesystemNode>> nodes)  
 Configure newly created nodes (e.g. setting another icon or changing the display name).

*sh::filesystem::FilesystemNodeType* **getType** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0  
 Determine if a file is regular, or a dir, or a link, ...

qint64 **getSize** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0  
 Determine the size of a file.

QDateTime **getMtime** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0  
 Determine the mtime of a file.

void **setMtime** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QDateTime time) = 0  
 Set the mtime of a file.

QString **getMimetype** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0  
 Determine mime type of a file.

QString **getLinkTarget** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0  
 Resolve a link.

QList<QString> **listExtendedAttributes** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)  
 Fetches the list of available extended attributes for an entry.

quint64 **getExtendedAttributeSize** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QString attribute)  
 Returns the size of value for one extended attribute for an entry.

QByteArray **getExtendedAttribute** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QString attribute)  
 Returns the value for one extended attribute for an entry.

void **setExtendedAttribute** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QString attribute, QByteArray value)  
 Sets the value for one extended attribute for an entry.

void **removeExtendedAttribute** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QString attribute)  
 Removes one extended attributes for an entry.

QMap<QString, QString> **getCustomAttributes** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)  
 Returns the custom attributes for an entry.

In contrast to extended attributes, the custom attributes are not directly stored in the filesystem this way, but handle implementation specific aspects (like permissions).

void **setCustomAttribute** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QString attribute, QString value)  
 Sets the value for one custom attribute for an entry.

See also getCustomAttributes.

bool **canGetFileContent** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0  
 Returns if it is allowed to get the content of a certain file.

```
std::shared_ptr<QIODevice> getFileContent (sh::filesystem::Operation *op,
                                             std::shared_ptr<const sh::filesystem::Eurl> eurl) =
                                             0
    Get the content of a file.

bool canCreateDirectory (sh::filesystem::Operation *op, std::shared_ptr<const
                        sh::filesystem::Eurl> eurl) = 0
    Returns if it is allowed to create subdirectories in a certain directory.

void createDirectory (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                      eurl) = 0
    Create a directory.

bool canCreateLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl) = 0
    Returns if it is allowed to create a link in a certain directory.

void createLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
                 QString target) = 0
    Create a link.

bool canCreateFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl) = 0
    Returns if it is allowed to create files in a certain directory.

void createFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
                 QIODevice *content, HandlerTransfer *handlertransfer) = 0
    Creates a file with some content.

    It may use handlertransfer for some better integration, like cancel support and progress monitoring.

bool canDeleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl) = 0
    Returns if it is allowed to delete a certain file/directory/link/...

void deleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl) =
    0
    Delete a file/directory/link/...

void deleteDirectoryIfEmpty (sh::filesystem::Operation *op, std::shared_ptr<const
                            sh::filesystem::Eurl> eurl)
    Deletes a directory only if it is empty.

bool canRenameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    src) = 0
    Returns if it is allowed to rename a certain file, directory, link, ... (see renameItem).

void renameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> src,
                 QString destpath) = 0
    Renames an item to another path.

    It can be a simple filename change or also changes in the deeper path segments.

QList<std::shared_ptr<sh::actions::AbstractActionItem>> getActions (const
                                                                QList<std::shared_ptr<const
                                                                sh::filesystem::Eurl>> eurls)
                                                                = 0
    Returns a list of actions (see former example) for showing for certain files.

bool requiresResolvedLinks ()
    Returns if this handler requires to be used by the engine with resolved links.

void viewEnteredDirectory (std::shared_ptr<const sh::filesystem::Eurl>)
    Callback for preparing stuff when the view entered some directory (e.g. for enabling watchers). See also
    viewLeftDirectory.
```

void **viewLeftDirectory** (std::shared\_ptr<const *sh::filesystem::Eurl*>)  
 Callback for preparing stuff when the view left some directory. See also viewEnteredDirectory.

bool **isWellKnownScheme** ()  
 Returns if the scheme for this handler is a well known one (which external programs typically would understand).

void **search** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, std::function<void> std::shared\_ptr<const *sh::filesystem::Eurl*>, QList<std::shared\_ptr<*sh::search::SearchCriterion*>>, *sh::filesystem::FilesystemNodeType* ftype  
 > addfile, std::function<void>std::shared\_ptr<const *sh::filesystem::Eurl*>> visitdir, QList<std::shared\_ptr<*sh::search::SearchCriterion*>> criteria)  
 Helps searching for files.

Override this method with a behavior identical to the default, if a specialized implementation can be much faster than the default one.

void **customizeUi** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> node, bool \*showSearchPanel)  
 Creates a custom gui, which becomes part of the fileview.

*sh::filesystem::FilesystemModel* \***model** ()  
 A convenience shortcut to the *sh::filesystem::FilesystemModel* (a singleton).

## Public Static Functions

void **doInitialize** ()  
 void **doShutdown** ()

### 10.1.186 Class *sh::filesystemhandlers::ArchiveFilesystemHandler::ActionOpenTarArchive*

```
class sh::filesystemhandlers::ArchiveFilesystemHandler::ActionOpenTarArchive : public sh::action
```

## Public Functions

**ActionOpenTarArchive** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> l)  
 QString **label** ()  
 QString **scheme** ()  
 void **execute** ()  
 Executes this action.  
 void **execute** (*sh::actions::ActionExecutionInfo* \*info)  
 Executes this action.  
 QKeySequence **shortcuthint** ()  
 Returns the keyboard shortcut for triggering this action.  
 bool **shortcuthintTriggersOnCurrentDirectory** ()  
 Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).  
 void **setShortcuthint** (QKeySequence shortcut, bool triggersOnCurrentDirectory = false)  
 Sets the keyboard shortcut for triggering this action.  
 QString **text** ()  
 Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<AbstractActionItem> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<AbstractActionItem> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.



## Signals

void **changed** ()  
Emits when some data changed.

### 10.1.187 Class `sh::filesystemhandlers::ArchiveFilesystemHandler::ActionOpenZipArchive`

```
class sh::filesystemhandlers::ArchiveFilesystemHandler::ActionOpenZipArchive : public sh::action
```

#### Public Functions

**ActionOpenZipArchive** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> *l*)

QString **label** ()

QString **scheme** ()

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<AbstractActionItem> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<AbstractActionItem> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0)  
Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

### 10.1.188 Class `sh::filesystemhandlers::GnomeIODevicesFilesystemHandler`

**class** `sh::filesystemhandlers::GnomeIODevicesFilesystemHandler` : **public** `sh::filesystemhandlers::GnomeIODevicesFilesystemHandler`  
A filesystem handler for the GIO filesystem ‘Devices’ subtree.

#### Public Functions

**~GnomeIODevicesFilesystemHandler** ()

void **itemlist** (`sh::filesystem::Operation` \**op*, std::shared\_ptr<const `sh::filesystem::Eurl`> *eurl*,  
`sh::filesystem::FilesystemNodeType` *type*, `sh::filesystem::FilesystemNodeListEditor`  
\**list*) **override**

`sh::filesystem::FilesystemNodeType` **getType** (`sh::filesystem::Operation` \**op*, std::shared\_ptr<const  
`sh::filesystem::Eurl`> *eurl*) **override**

QList<std::shared\_ptr<`sh::actions::AbstractActionItem`>> **getActions** (const  
QList<std::shared\_ptr<const  
`sh::filesystem::Eurl`>> *eurls*)  
**override**

void **refresh** ()  
Refreshes the device list.

void **requestNewNode** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*,  
std::shared\_ptr<*sh::filesystem::FilesystemNode*> \**nnode*)  
Requests a new device node.

void **itemlist** (*sh::filesystem::Operation* \**op*, std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*,  
*sh::filesystem::FilesystemNodeType* *type*, *sh::filesystem::FilesystemNodeListEditor*  
\**list*) = 0  
Determine a list of subelements in a certain directory.

*sh::filesystem::FilesystemNodeType* **getType** (*sh::filesystem::Operation* \**op*, std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*) = 0  
Determine if a file is regular, or a dir, or a link, ...

qint64 **getSize** (*sh::filesystem::Operation* \**op*, std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*)  
qint64 **getSize** (*sh::filesystem::Operation* \**op*, std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*) = 0  
Determine the size of a file.

QDateTime **getMtime** (*sh::filesystem::Operation* \**op*, std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*)  
QDateTime **getMtime** (*sh::filesystem::Operation* \**op*, std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*) = 0  
Determine the mtime of a file.

void **setMtime** (*sh::filesystem::Operation* \**op*, std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*,  
QDateTime *time*)  
void **setMtime** (*sh::filesystem::Operation* \**op*, std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*,  
QDateTime *time*) = 0  
Set the mtime of a file.

QString **getMimeType** (*sh::filesystem::Operation* \**op*, std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*)  
QString **getMimeType** (*sh::filesystem::Operation* \**op*, std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*) = 0  
Determine mime type of a file.

QString **getLinkTarget** (*sh::filesystem::Operation* \**op*, std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*)  
QString **getLinkTarget** (*sh::filesystem::Operation* \**op*, std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*) = 0  
Resolve a link.

bool **canGetFileContent** (*sh::filesystem::Operation* \**op*, std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*)  
bool **canGetFileContent** (*sh::filesystem::Operation* \**op*, std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*) = 0  
Returns if it is allowed to get the content of a certain file.

std::shared\_ptr<QIODevice> **getFileContent** (*sh::filesystem::Operation* \**op*, std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*)  
std::shared\_ptr<QIODevice> **getFileContent** (*sh::filesystem::Operation* \**op*, std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*) = 0  
Get the content of a file.

bool **canCreateDirectory** (*sh::filesystem::Operation* \**op*, std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*)  
bool **canCreateDirectory** (*sh::filesystem::Operation* \**op*, std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*) = 0  
Returns if it is allowed to create subdirectories in a certain directory.

```
void createDirectory (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl)
```

```
void createDirectory (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl) = 0
    Create a directory.
```

```
bool canCreateLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                   eurl)
```

```
bool canCreateLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                   eurl) = 0
    Returns if it is allowed to create a link in a certain directory.
```

```
void createLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
                QString target)
```

```
void createLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
                QString target) = 0
    Create a link.
```

```
bool canCreateFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                   eurl)
```

```
bool canCreateFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                   eurl) = 0
    Returns if it is allowed to create files in a certain directory.
```

```
void createFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
                QIODevice *content, HandlerTransfer *handlertransfer)
```

```
void createFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
                QIODevice *content, HandlerTransfer *handlertransfer) = 0
    Creates a file with some content.
```

It may use `handlertransfer` for some better integration, like cancel support and progress monitoring.

```
bool canDeleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                   eurl)
```

```
bool canDeleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                   eurl) = 0
    Returns if it is allowed to delete a certain file/directory/link/...
```

```
void deleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl)
```

```
void deleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl) =
    0
    Delete a file/directory/link/...
```

```
bool canRenameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                   src)
```

```
bool canRenameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                   src) = 0
    Returns if it is allowed to rename a certain file, directory, link, ... (see renameItem).
```

```
void renameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> src,
                QString destpath)
```

```
void renameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> src,
                QString destpath) = 0
    Renames an item to another path.
```

It can be a simple filename change or also changes in the deeper path segments.

`QList<std::shared_ptr<sh::actions::AbstractActionItem>> getActions (const  
QList<std::shared_ptr<const  
sh::filesystem::Eurl>> eurls)  
= 0`  
Returns a list of actions (see former example) for showing for certain files.

`bool requiresResolvedLinks ()`  
Returns if this handler requires to be used by the engine with resolved links.

`bool isWellKnownScheme ()`  
Returns if the scheme for this handler is a well known one (which external programs typically would understand).

`void configureItems (sh::filesystem::Operation *op, QList<std::shared_ptr<FilesystemNode>>  
nodes)`  
Configure newly created nodes (e.g. setting another icon or changing the display name).

`QList<QString> listExtendedAttributes (sh::filesystem::Operation *op, std::shared_ptr<const  
sh::filesystem::Eurl> eurl)`  
Fetches the list of available extended attributes for an entry.

`quint64 getExtendedAttributesSize (sh::filesystem::Operation *op, std::shared_ptr<const  
sh::filesystem::Eurl> eurl, QString attribute)`  
Returns the size of value for one extended attribute for an entry.

`QByteArray getExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const  
sh::filesystem::Eurl> eurl, QString attribute)`  
Returns the value for one extended attribute for an entry.

`void setExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const  
sh::filesystem::Eurl> eurl, QString attribute, QByteArray value)`  
Sets the value for one extended attribute for an entry.

`void removeExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const  
sh::filesystem::Eurl> eurl, QString attribute)`  
Removes one extended attributes for an entry.

`QMap<QString, QString> getCustomAttributes (sh::filesystem::Operation *op,  
std::shared_ptr<const sh::filesystem::Eurl>  
eurl)`  
Returns the custom attributes for an entry.

In contrast to extended attributes, the custom attributes are not directly stored in the filesystem this way, but handle implementation specific aspects (like permissions).

`void setCustomAttribute (sh::filesystem::Operation *op, std::shared_ptr<const  
sh::filesystem::Eurl> eurl, QString attribute, QString value)`  
Sets the value for one custom attribute for an entry.

See also `getCustomAttributes`.

`void deleteDirectoryIfEmpty (sh::filesystem::Operation *op, std::shared_ptr<const  
sh::filesystem::Eurl> eurl)`  
Deletes a directory only if it is empty.

`void viewEnteredDirectory (std::shared_ptr<const sh::filesystem::Eurl>)`  
Callback for preparing stuff when the view entered some directory (e.g. for enabling watchers). See also `viewLeftDirectory`.

`void viewLeftDirectory (std::shared_ptr<const sh::filesystem::Eurl>)`  
Callback for preparing stuff when the view left some directory. See also `viewEnteredDirectory`.

```
void search (sh::filesystem::Operation *op, std::shared_ptr<const  
    sh::filesystem::Eurl> eurl, std::function<void> std::shared_ptr<const  
    sh::filesystem::Eurl>, QList<std::shared_ptr<sh::search::SearchCriterion>>,  
    sh::filesystem::FilesystemNodeType ftype  
> addfile, std::function<void>std::shared_ptr<const sh::filesystem::Eurl>> visitdir,  
    QList<std::shared_ptr<sh::search::SearchCriterion>> criteria)Helps searching for files.
```

Override this method with a behavior identical to the default, if a specialized implementation can be much faster than the default one.

```
void customizeUi (std::shared_ptr<sh::filesystem::FilesystemNode> node, bool *showSearchPanel)  
    Creates a custom gui, which becomes part of the fileview.
```

```
sh::filesystem::FilesystemModel *model ()  
    A convenience shortcut to the sh::filesystem::FilesystemModel (a singleton).
```

```
void doShutdown ()  
    Executes singleton shutdown.
```

```
void shutdown ()  
    Shutdown down this singleton.
```

```
bool isAlive ()  
    Returns if this singleton is alive (true until its shutdown begins).
```

## Public Static Functions

```
QString mountDevice (QString gvolumeuuid, sh::actions::ActionExecutionInfo *info)
```

```
QString mountLocation (QString gurl, sh::actions::ActionExecutionInfo *info)
```

```
QString unmountLocation (QString gurl, sh::actions::ActionExecutionInfo *info)
```

```
QByteArray eurl2gurlb (std::shared_ptr<const sh::filesystem::Eurl> eurl)
```

```
QString eurl2gurl (std::shared_ptr<const sh::filesystem::Eurl> eurl)
```

```
std::shared_ptr<const sh::filesystem::Eurl> gurl2eurl (QString gurl)
```

```
void logIfError (void *error)
```

```
void throwIfError (void *error)
```

## Private Functions

```
GnomeIODevicesFilesystemHandler ()
```

## Friends

```
friend class ActionMount
```

### 10.1.189 Class `sh::filesystemhandlers::GnomeIOFilesystemHandler`

**class** `sh::filesystemhandlers::GnomeIOFilesystemHandler` : public `sh::filesystem::FilesystemHandler`  
 A filesystem handler for GIO filesystem parts.

This is the abstract base class.

Subclassed by `sh::filesystemhandlers::GnomeIODevicesFilesystemHandler`,  
`sh::filesystemhandlers::GnomeIOGenericFilesystemHandler`, `sh::filesystemhandlers::GnomeIONetworkFilesystemHandler`,  
`sh::filesystemhandlers::GnomeIOSmbFilesystemHandler`

#### Public Functions

**GnomeIOFilesystemHandler()**

void **itemlist** (`sh::filesystem::Operation` \*op, std::shared\_ptr<const `sh::filesystem::Eurl`> eurl,  
`sh::filesystem::FilesystemNodeType` type, `sh::filesystem::FilesystemNodeListEditor`  
 \*list)

`sh::filesystem::FilesystemNodeType` **getType** (`sh::filesystem::Operation` \*op, std::shared\_ptr<const  
`sh::filesystem::Eurl`> eurl)

qint64 **getSize** (`sh::filesystem::Operation` \*op, std::shared\_ptr<const `sh::filesystem::Eurl`> eurl)

QDateTime **getMtime** (`sh::filesystem::Operation` \*op, std::shared\_ptr<const `sh::filesystem::Eurl`>  
 eurl)

void **setMtime** (`sh::filesystem::Operation` \*op, std::shared\_ptr<const `sh::filesystem::Eurl`> eurl,  
 QDateTime time)

QString **getMimeType** (`sh::filesystem::Operation` \*op, std::shared\_ptr<const `sh::filesystem::Eurl`>  
 eurl)

QString **getLinkTarget** (`sh::filesystem::Operation` \*op, std::shared\_ptr<const `sh::filesystem::Eurl`>  
 eurl)

bool **canGetFileContent** (`sh::filesystem::Operation` \*op, std::shared\_ptr<const  
`sh::filesystem::Eurl`> eurl)

std::shared\_ptr<QIODevice> **getFileContent** (`sh::filesystem::Operation` \*op,  
 std::shared\_ptr<const `sh::filesystem::Eurl`> eurl)

bool **canCreateDirectory** (`sh::filesystem::Operation` \*op, std::shared\_ptr<const  
`sh::filesystem::Eurl`> eurl)

void **createDirectory** (`sh::filesystem::Operation` \*op, std::shared\_ptr<const `sh::filesystem::Eurl`>  
 eurl)

bool **canCreateLink** (`sh::filesystem::Operation` \*op, std::shared\_ptr<const `sh::filesystem::Eurl`>  
 eurl)

void **createLink** (`sh::filesystem::Operation` \*op, std::shared\_ptr<const `sh::filesystem::Eurl`> eurl,  
 QString target)

bool **canCreateFile** (`sh::filesystem::Operation` \*op, std::shared\_ptr<const `sh::filesystem::Eurl`>  
 eurl)

void **createFile** (`sh::filesystem::Operation` \*op, std::shared\_ptr<const `sh::filesystem::Eurl`> eurl,  
 QIODevice \*content, HandlerTransfer \*handlertransfer)

bool **canDeleteItem** (`sh::filesystem::Operation` \*op, std::shared\_ptr<const `sh::filesystem::Eurl`>  
 eurl)

void **deleteItem** (`sh::filesystem::Operation` \*op, std::shared\_ptr<const `sh::filesystem::Eurl`> eurl)

bool **canRenameItem** (`sh::filesystem::Operation` \*op, std::shared\_ptr<const `sh::filesystem::Eurl`>  
 src)



```
void renameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> src,
                QString destpath)

QList<std::shared_ptr<sh::actions::AbstractActionItem>> getActions (const
                                                                QList<std::shared_ptr<const
                                                                sh::filesystem::Eurl>> eurls)

bool requiresResolvedLinks ()
    Returns if this handler requires to be used by the engine with resolved links.

bool isWellKnownScheme ()
    Returns if the scheme for this handler is a well known one (which external programs typically would
    understand).

void itemlist (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
              sh::filesystem::FileSystemNodeType type, sh::filesystem::FileSystemNodeListEditor
              *list) = 0
    Determine a list of subelements in a certain directory.

void configureItems (sh::filesystem::Operation *op, QList<std::shared_ptr<FileSystemNode>>
                    nodes)
    Configure newly created nodes (e.g. setting another icon or changing the display name).

sh::filesystem::FileSystemNodeType getType (sh::filesystem::Operation *op, std::shared_ptr<const
                                           sh::filesystem::Eurl> eurl) = 0
    Determine if a file is regular, or a dir, or a link, ...

qint64 getSize (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl) = 0
    Determine the size of a file.

QDateTime getMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl) = 0
    Determine the mtime of a file.

void setMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
              QDateTime time) = 0
    Set the mtime of a file.

QString getMimetype (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl) = 0
    Determine mime type of a file.

QString getLinkTarget (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl) = 0
    Resolve a link.

QList<QString> listExtendedAttributes (sh::filesystem::Operation *op, std::shared_ptr<const
                                      sh::filesystem::Eurl> eurl)
    Fetches the list of available extended attributes for an entry.

quint64 getExtendedAttributeSize (sh::filesystem::Operation *op, std::shared_ptr<const
                                   sh::filesystem::Eurl> eurl, QString attribute)
    Returns the size of value for one extended attribute for an entry.

QByteArray getExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
                                   sh::filesystem::Eurl> eurl, QString attribute)
    Returns the value for one extended attribute for an entry.

void setExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
                           sh::filesystem::Eurl> eurl, QString attribute, QByteArray value)
    Sets the value for one extended attribute for an entry.

void removeExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
                              sh::filesystem::Eurl> eurl, QString attribute)
    Removes one extended attributes for an entry.
```



```
QMap<QString, QString> getCustomAttributes (sh::filesystem::Operation *op,
                                              std::shared_ptr<const sh::filesystem::Eurl>
                                              eurl)
```

Returns the custom attributes for an entry.

In contrast to extended attributes, the custom attributes are not directly stored in the filesystem this way, but handle implementation specific aspects (like permissions).

```
void setCustomAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
                        sh::filesystem::Eurl> eurl, QString attribute, QString value)
```

Sets the value for one custom attribute for an entry.

See also getCustomAttributes.

```
bool canGetFileContent (sh::filesystem::Operation *op, std::shared_ptr<const
                        sh::filesystem::Eurl> eurl) = 0
```

Returns if it is allowed to get the content of a certain file.

```
std::shared_ptr<QIODevice> getFileContent (sh::filesystem::Operation *op,
                                             std::shared_ptr<const sh::filesystem::Eurl> eurl) =
0
```

Get the content of a file.

```
bool canCreateDirectory (sh::filesystem::Operation *op, std::shared_ptr<const
                        sh::filesystem::Eurl> eurl) = 0
```

Returns if it is allowed to create subdirectories in a certain directory.

```
void createDirectory (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                      eurl) = 0
```

Create a directory.

```
bool canCreateLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl) = 0
```

Returns if it is allowed to create a link in a certain directory.

```
void createLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
                 QString target) = 0
```

Create a link.

```
bool canCreateFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl) = 0
```

Returns if it is allowed to create files in a certain directory.

```
void createFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
                 QIODevice *content, HandlerTransfer *handlertransfer) = 0
```

Creates a file with some content.

It may use handlertransfer for some better integration, like cancel support and progress monitoring.

```
bool canDeleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl) = 0
```

Returns if it is allowed to delete a certain file/directory/link/...

```
void deleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl) =
0
```

Delete a file/directory/link/...

```
void deleteDirectoryIfEmpty (sh::filesystem::Operation *op, std::shared_ptr<const
                             sh::filesystem::Eurl> eurl)
```

Deletes a directory only if it is empty.

```
bool canRenameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    src) = 0
```

Returns if it is allowed to rename a certain file, directory, link, ... (see renameItem).

void **renameItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> src, QString destpath) = 0  
Renames an item to another path.

It can be a simple filename change or also changes in the deeper path segments.

QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> **getActions** (const QList<std::shared\_ptr<const *sh::filesystem::Eurl*>> eurls)  
= 0

Returns a list of actions (see former example) for showing for certain files.

void **viewEnteredDirectory** (std::shared\_ptr<const *sh::filesystem::Eurl*>)  
Callback for preparing stuff when the view entered some directory (e.g. for enabling watchers). See also viewLeftDirectory.

void **viewLeftDirectory** (std::shared\_ptr<const *sh::filesystem::Eurl*>)  
Callback for preparing stuff when the view left some directory. See also viewEnteredDirectory.

void **search** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, std::function<void> std::shared\_ptr<const *sh::filesystem::Eurl*>, QList<std::shared\_ptr<*sh::search::SearchCriterion*>>, *sh::filesystem::FilesystemNodeType* ftype  
> addfile, std::function<void> std::shared\_ptr<const *sh::filesystem::Eurl*>> visitdir, QList<std::shared\_ptr<*sh::search::SearchCriterion*>> criteria)  
Helps searching for files.

Override this method with a behavior identical to the default, if a specialized implementation can be much faster than the default one.

void **customizeUi** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> node, bool \*showSearchPanel)  
Creates a custom gui, which becomes part of the fileview.

*sh::filesystem::FilesystemModel* \***model** ()  
A convenience shortcut to the *sh::filesystem::FilesystemModel* (a singleton).

## Public Static Functions

QString **mountDevice** (QString gvolumeuuid, *sh::actions::ActionExecutionInfo* \*info)

QString **mountLocation** (QString gurl, *sh::actions::ActionExecutionInfo* \*info)

QString **unmountLocation** (QString gurl, *sh::actions::ActionExecutionInfo* \*info)

QByteArray **eurl2gurlb** (std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)

QString **eurl2gurl** (std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)

std::shared\_ptr<const *sh::filesystem::Eurl*> **gurl2eurl** (QString gurl)

void **logIfError** (void \*error)

void **throwIfError** (void \*error)

**class MountOperations**

**Public Static Functions**

```
void *createMountOperation (sh::actions::ActionExecutionInfo *info, QString address,
                           MountOperationsCallbackData **opdata)
void deleteMountOperation (MountOperationsCallbackData *opdata)
```

**Public Static Attributes**

```
QHash<size_t, MountOperationsCallbackData*> callbackdata
size_t callbackdataindex
QMutex callbackdatamutex
struct MountOperationsCallbackData
```

**Public Functions**

```
MountOperationsCallbackData (sh::actions::ActionExecutionInfo *info, size_t index, void
                             *mountoperation, QString address)
```

**Public Members**

```
sh::actions::ActionExecutionInfo *info
size_t index
void *mountoperation
QString address
```

**10.1.190 Class sh::filesystemhandlers::GnomeIOFilesystemHandler::GnomeIOReadDataDevice**

```
class sh::filesystemhandlers::GnomeIOFilesystemHandler::GnomeIOReadDataDevice : public QIODevice
```

**Public Functions**

```
GnomeIOReadDataDevice (QString guri)
~GnomeIOReadDataDevice ()
```

**10.1.191 Class sh::filesystemhandlers::GnomeIOFilesystemHandler::MountOperations**

```
class sh::filesystemhandlers::GnomeIOFilesystemHandler::MountOperations
```

## Public Static Functions

```
void createMountOperation (sh::actions::ActionExecutionInfo *info, QString address, MountOperationsCallbackData **opdata)

void deleteMountOperation (MountOperationsCallbackData *opdata)
```

## Public Static Attributes

```
QHash<size_t, MountOperationsCallbackData*> callbackdata

size_t callbackdataindex

QMutex callbackdatamutex
```

### 10.1.192 Class *sh::filesystemhandlers::GnomeIOGenericFilesystemHandler*

```
class sh::filesystemhandlers::GnomeIOGenericFilesystemHandler : public sh::filesystemhandlers::GnomeIO
```

A filesystem handler for common/unspecial GIO filesystem parts.

This is a generic non-abstract implementation without special behavior.

## Public Functions

```
GnomeIOGenericFilesystemHandler (sh::filesystem::FilesystemModel *model)
```

```
void itemlist (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
               sh::filesystem::FilesystemNodeType type, sh::filesystem::FilesystemNodeListEditor
               *list)
```

```
void itemlist (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
               sh::filesystem::FilesystemNodeType type, sh::filesystem::FilesystemNodeListEditor
               *list) = 0
```

Determine a list of subelements in a certain directory.

```
sh::filesystem::FilesystemNodeType getType (sh::filesystem::Operation *op, std::shared_ptr<const
                                             sh::filesystem::Eurl> eurl)
```

```
sh::filesystem::FilesystemNodeType getType (sh::filesystem::Operation *op, std::shared_ptr<const
                                             sh::filesystem::Eurl> eurl) = 0
```

Determine if a file is regular, or a dir, or a link, ...

```
qint64 getSize (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl)
```

```
qint64 getSize (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl) = 0
```

Determine the size of a file.

```
QDateTime getMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl)
```

```
QDateTime getMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl) = 0
```

Determine the mtime of a file.

```
void setMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
               QDateTime time)
```

```
void setMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
               QDateTime time) = 0
```

Set the mtime of a file.

```

QString getMimeType (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl)
QString getMimeType (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl) = 0
    Determine mime type of a file.

QString getLinkTarget (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl)
QString getLinkTarget (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl) = 0
    Resolve a link.

bool canGetFileContent (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl)
bool canGetFileContent (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) = 0
    Returns if it is allowed to get the content of a certain file.

std::shared_ptr<QIODevice> getFileContent (sh::filesystem::Operation *op,
                    std::shared_ptr<const sh::filesystem::Eurl> eurl)
std::shared_ptr<QIODevice> getFileContent (sh::filesystem::Operation *op,
                    std::shared_ptr<const sh::filesystem::Eurl> eurl) =
    0
    Get the content of a file.

bool canCreateDirectory (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl)
bool canCreateDirectory (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) = 0
    Returns if it is allowed to create subdirectories in a certain directory.

void createDirectory (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl)
void createDirectory (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl) = 0
    Create a directory.

bool canCreateLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl)
bool canCreateLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl) = 0
    Returns if it is allowed to create a link in a certain directory.

void createLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
                    QString target)
void createLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
                    QString target) = 0
    Create a link.

bool canCreateFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl)
bool canCreateFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl) = 0
    Returns if it is allowed to create files in a certain directory.

void createFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
                    QIODevice *content, HandlerTransfer *handlertransfer)

```

void **createFile** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QIODevice \*content, HandlerTransfer \*handlertransfer) = 0  
Creates a file with some content.

It may use handlertransfer for some better integration, like cancel support and progress monitoring.

bool **canDeleteItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)

bool **canDeleteItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0

Returns if it is allowed to delete a certain file/directory/link/...

void **deleteItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)

void **deleteItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0

Delete a file/directory/link/...

bool **canRenameItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> src)

bool **canRenameItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> src) = 0

Returns if it is allowed to rename a certain file, directory, link, ... (see renameItem).

void **renameItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> src, QString destpath)

void **renameItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> src, QString destpath) = 0

Renames an item to another path.

It can be a simple filename change or also changes in the deeper path segments.

QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> **getActions** (const QList<std::shared\_ptr<const *sh::filesystem::Eurl*>> eurls)

QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> **getActions** (const QList<std::shared\_ptr<const *sh::filesystem::Eurl*>> eurls) = 0

Returns a list of actions (see former example) for showing for certain files.

bool **requiresResolvedLinks** ()

Returns if this handler requires to be used by the engine with resolved links.

bool **isWellKnownScheme** ()

Returns if the scheme for this handler is a well known one (which external programs typically would understand).

void **configureItems** (*sh::filesystem::Operation* \*op, QList<std::shared\_ptr<FileSystemNode>> nodes)

Configure newly created nodes (e.g. setting another icon or changing the display name).

QList<QString> **listExtendedAttributes** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)

Fetches the list of available extended attributes for an entry.

quint64 **getExtendedAttributeSize** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QString attribute)

Returns the size of value for one extended attribute for an entry.

QByteArray **getExtendedAttribute** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QString attribute)

Returns the value for one extended attribute for an entry.

void **setExtendedAttribute** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QString attribute, QByteArray value)

Sets the value for one extended attribute for an entry.

void **removeExtendedAttribute** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QString attribute)

Removes one extended attributes for an entry.

QMap<QString, QString> **getCustomAttributes** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)

Returns the custom attributes for an entry.

In contrast to extended attributes, the custom attributes are not directly stored in the filesystem this way, but handle implementation specific aspects (like permissions).

void **setCustomAttribute** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QString attribute, QString value)

Sets the value for one custom attribute for an entry.

See also getCustomAttributes.

void **deleteDirectoryIfEmpty** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)

Deletes a directory only if it is empty.

void **viewEnteredDirectory** (std::shared\_ptr<const *sh::filesystem::Eurl*>)

Callback for preparing stuff when the view entered some directory (e.g. for enabling watchers). See also viewLeftDirectory.

void **viewLeftDirectory** (std::shared\_ptr<const *sh::filesystem::Eurl*>)

Callback for preparing stuff when the view left some directory. See also viewEnteredDirectory.

void **search** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, std::function<void> std::shared\_ptr<const *sh::filesystem::Eurl*>, QList<std::shared\_ptr<*sh::search::SearchCriterion*>>, *sh::filesystem::FilesystemNodeType* ftype > addfile, std::function<void> std::shared\_ptr<const *sh::filesystem::Eurl*>> visitdir, QList<std::shared\_ptr<*sh::search::SearchCriterion*>> criteria) Helps searching for files.

Override this method with a behavior identical to the default, if a specialized implementation can be much faster than the default one.

void **customizeUi** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> node, bool \*showSearchPanel)

Creates a custom gui, which becomes part of the fileview.

*sh::filesystem::FilesystemModel* \*model ()

A convenience shortcut to the *sh::filesystem::FilesystemModel* (a singleton).



## Public Static Functions

```
QString mountDevice (QString gvolumeuuid, sh::actions::ActionExecutionInfo *info)
QString mountLocation (QString gurl, sh::actions::ActionExecutionInfo *info)
QString unmountLocation (QString gurl, sh::actions::ActionExecutionInfo *info)
QByteArray eurl2gurlb (std::shared_ptr<const sh::filesystem::Eurl> eurl)
QString eurl2gurl (std::shared_ptr<const sh::filesystem::Eurl> eurl)
std::shared_ptr<const sh::filesystem::Eurl> gurl2eurl (QString gurl)
void logIfError (void *error)
void throwIfError (void *error)
```

### 10.1.193 Class sh::filesystemhandlers::GnomeIONetworkFilesystemHandler

```
class sh::filesystemhandlers::GnomeIONetworkFilesystemHandler : public sh::filesystemhandlers::GnomeIO
    A filesystem handler for the GIO filesystem 'Network' subtree.
```

## Public Functions

```
QList<std::shared_ptr<sh::actions::AbstractActionItem>> getActions (const
                                                                    QList<std::shared_ptr<const
                                                                    sh::filesystem::Eurl>> eurls)
                                                                    override
void requestNewNode (std::shared_ptr<const sh::filesystem::Eurl> eurl,
                    std::shared_ptr<sh::filesystem::FilesystemNode> *nnode)
    Requests a new network node.
void itemList (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
              sh::filesystem::FilesystemNodeType type, sh::filesystem::FilesystemNodeListEditor
              *list)
void itemList (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
              sh::filesystem::FilesystemNodeType type, sh::filesystem::FilesystemNodeListEditor
              *list) = 0
    Determine a list of subelements in a certain directory.
sh::filesystem::FilesystemNodeType getType (sh::filesystem::Operation *op, std::shared_ptr<const
                                           sh::filesystem::Eurl> eurl)
sh::filesystem::FilesystemNodeType getType (sh::filesystem::Operation *op, std::shared_ptr<const
                                           sh::filesystem::Eurl> eurl) = 0
    Determine if a file is regular, or a dir, or a link, ...
qint64 getSize (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl)
qint64 getSize (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl) = 0
    Determine the size of a file.
QDateTime getMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                   eurl)
QDateTime getMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                   eurl) = 0
    Determine the mtime of a file.
```



```

void setMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
               QDateTime time)
void setMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
               QDateTime time) = 0
    Set the mtime of a file.

QString getMimeType (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl)
QString getMimeType (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl) = 0
    Determine mime type of a file.

QString getLinkTarget (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl)
QString getLinkTarget (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl) = 0
    Resolve a link.

bool canGetFileContent (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl)
bool canGetFileContent (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) = 0
    Returns if it is allowed to get the content of a certain file.

std::shared_ptr<QIODevice> getFileContent (sh::filesystem::Operation *op,
                                           std::shared_ptr<const sh::filesystem::Eurl> eurl)
std::shared_ptr<QIODevice> getFileContent (sh::filesystem::Operation *op,
                                           std::shared_ptr<const sh::filesystem::Eurl> eurl) =
    0
    Get the content of a file.

bool canCreateDirectory (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl)
bool canCreateDirectory (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) = 0
    Returns if it is allowed to create subdirectories in a certain directory.

void createDirectory (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl)
void createDirectory (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl) = 0
    Create a directory.

bool canCreateLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl)
bool canCreateLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl) = 0
    Returns if it is allowed to create a link in a certain directory.

void createLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
                 QString target)
void createLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
                 QString target) = 0
    Create a link.

bool canCreateFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl)

```

bool **canCreateFile** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0

Returns if it is allowed to create files in a certain directory.

void **createFile** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QIODevice \*content, HandlerTransfer \*handlertransfer)

void **createFile** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QIODevice \*content, HandlerTransfer \*handlertransfer) = 0

Creates a file with some content.

It may use handlertransfer for some better integration, like cancel support and progress monitoring.

bool **canDeleteItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)

bool **canDeleteItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0

Returns if it is allowed to delete a certain file/directory/link/...

void **deleteItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)

void **deleteItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0

Delete a file/directory/link/...

bool **canRenameItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> src)

bool **canRenameItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> src) = 0

Returns if it is allowed to rename a certain file, directory, link, ... (see renameItem).

void **renameItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> src, QString destpath)

void **renameItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> src, QString destpath) = 0

Renames an item to another path.

It can be a simple filename change or also changes in the deeper path segments.

QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> **getActions** (const QList<std::shared\_ptr<const *sh::filesystem::Eurl*>> eurls) = 0

Returns a list of actions (see former example) for showing for certain files.

bool **requiresResolvedLinks** ()

Returns if this handler requires to be used by the engine with resolved links.

bool **isWellKnownScheme** ()

Returns if the scheme for this handler is a well known one (which external programs typically would understand).

void **configureItems** (*sh::filesystem::Operation* \*op, QList<std::shared\_ptr<FileSystemNode>> nodes)

Configure newly created nodes (e.g. setting another icon or changing the display name).

QList<QString> **listExtendedAttributes** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)

Fetches the list of available extended attributes for an entry.

quint64 **getExtendedAttributeSize** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QString attribute)

Returns the size of value for one extended attribute for an entry.

`QByteArray` **getExtendedAttribute** (*sh::filesystem::Operation* \*op, `std::shared_ptr<const sh::filesystem::Eurl>` eurl, `QString` attribute)  
Returns the value for one extended attribute for an entry.

void **setExtendedAttribute** (*sh::filesystem::Operation* \*op, `std::shared_ptr<const sh::filesystem::Eurl>` eurl, `QString` attribute, `QByteArray` value)  
Sets the value for one extended attribute for an entry.

void **removeExtendedAttribute** (*sh::filesystem::Operation* \*op, `std::shared_ptr<const sh::filesystem::Eurl>` eurl, `QString` attribute)  
Removes one extended attributes for an entry.

`QMap<QString, QString>` **getCustomAttributes** (*sh::filesystem::Operation* \*op, `std::shared_ptr<const sh::filesystem::Eurl>` eurl)  
Returns the custom attributes for an entry.

In contrast to extended attributes, the custom attributes are not directly stored in the filesystem this way, but handle implementation specific aspects (like permissions).

void **setCustomAttribute** (*sh::filesystem::Operation* \*op, `std::shared_ptr<const sh::filesystem::Eurl>` eurl, `QString` attribute, `QString` value)  
Sets the value for one custom attribute for an entry.

See also `getCustomAttributes`.

void **deleteDirectoryIfEmpty** (*sh::filesystem::Operation* \*op, `std::shared_ptr<const sh::filesystem::Eurl>` eurl)  
Deletes a directory only if it is empty.

void **viewEnteredDirectory** (`std::shared_ptr<const sh::filesystem::Eurl>`)  
Callback for preparing stuff when the view entered some directory (e.g. for enabling watchers). See also `viewLeftDirectory`.

void **viewLeftDirectory** (`std::shared_ptr<const sh::filesystem::Eurl>`)  
Callback for preparing stuff when the view left some directory. See also `viewEnteredDirectory`.

void **search** (*sh::filesystem::Operation* \*op, `std::shared_ptr<const sh::filesystem::Eurl>` eurl, `std::function<void>` std::shared\_ptr<const sh::filesystem::Eurl>, `QList<std::shared_ptr<sh::search::SearchCriterion>>`, *sh::filesystem::FilesystemNodeType* ftype  
> `addfile`, `std::function<void>`std::shared\_ptr<const sh::filesystem::Eurl>> `visitdir`, `QList<std::shared_ptr<sh::search::SearchCriterion>>` criteria)  
Helps searching for files.

Override this method with a behavior identical to the default, if a specialized implementation can be much faster than the default one.

void **customizeUi** (`std::shared_ptr<sh::filesystem::FilesystemNode>` node, `bool` \*showSearchPanel)  
Creates a custom gui, which becomes part of the fileview.

*sh::filesystem::FilesystemModel* \***model** ()  
A convenience shortcut to the *sh::filesystem::FilesystemModel* (a singleton).

void **doShutdown** ()  
Executes singleton shutdown.

void **shutdown** ()  
Shutdown down this singleton.

bool **isAlive** ()  
Returns if this singleton is alive (true until its shutdown begins).

## Public Static Functions

```
QString mountDevice (QString gvolumeuuid, sh::actions::ActionExecutionInfo *info)
QString mountLocation (QString gurl, sh::actions::ActionExecutionInfo *info)
QString unmountLocation (QString gurl, sh::actions::ActionExecutionInfo *info)
QByteArray eurl2gurlb (std::shared_ptr<const sh::filesystem::Eurl> eurl)
QString eurl2gurl (std::shared_ptr<const sh::filesystem::Eurl> eurl)
std::shared_ptr<const sh::filesystem::Eurl> gurl2eurl (QString gurl)
void logIfError (void *error)
void throwIfError (void *error)
```

## Private Functions

```
GnomeIONetworkFilesystemHandler ()
```

### 10.1.194 Class sh::filesystemhandlers::GnomeIOSmbFilesystemHandler

```
class sh::filesystemhandlers::GnomeIOSmbFilesystemHandler : public sh::filesystemhandlers::GnomeIOFiles
    A filesystem handler for the GIO filesystem ‘smb’ subtrees (in ‘Network’).
```

## Public Functions

```
sh::filesystem::FilesystemNodeType getType (sh::filesystem::Operation *op, std::shared_ptr<const
                                           sh::filesystem::Eurl> eurl) override
void itemList (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
               sh::filesystem::FilesystemNodeType type, sh::filesystem::FilesystemNodeListEditor
               *list)
void itemList (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
               sh::filesystem::FilesystemNodeType type, sh::filesystem::FilesystemNodeListEditor
               *list) = 0
    Determine a list of subelements in a certain directory.
sh::filesystem::FilesystemNodeType getType (sh::filesystem::Operation *op, std::shared_ptr<const
                                           sh::filesystem::Eurl> eurl) = 0
    Determine if a file is regular, or a dir, or a link, ...
qint64 getSize (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl)
qint64 getSize (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl) = 0
    Determine the size of a file.
QDateTime getMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl)
QDateTime getMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl) = 0
    Determine the mtime of a file.
void setMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
               QDateTime time)
```

```

void setMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
               QDateTime time) = 0
    Set the mtime of a file.

QString getMimeType (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                     eurl)

QString getMimeType (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                     eurl) = 0
    Determine mime type of a file.

QString getLinkTarget (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                       eurl)

QString getLinkTarget (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                       eurl) = 0
    Resolve a link.

bool canGetFileContent (sh::filesystem::Operation *op, std::shared_ptr<const
                       sh::filesystem::Eurl> eurl)

bool canGetFileContent (sh::filesystem::Operation *op, std::shared_ptr<const
                       sh::filesystem::Eurl> eurl) = 0
    Returns if it is allowed to get the content of a certain file.

std::shared_ptr<QIODevice> getFileContent (sh::filesystem::Operation *op,
                                           std::shared_ptr<const sh::filesystem::Eurl> eurl)

std::shared_ptr<QIODevice> getFileContent (sh::filesystem::Operation *op,
                                           std::shared_ptr<const sh::filesystem::Eurl> eurl) =
    0
    Get the content of a file.

bool canCreateDirectory (sh::filesystem::Operation *op, std::shared_ptr<const
                       sh::filesystem::Eurl> eurl)

bool canCreateDirectory (sh::filesystem::Operation *op, std::shared_ptr<const
                       sh::filesystem::Eurl> eurl) = 0
    Returns if it is allowed to create subdirectories in a certain directory.

void createDirectory (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                     eurl)

void createDirectory (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                     eurl) = 0
    Create a directory.

bool canCreateLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl)

bool canCreateLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl) = 0
    Returns if it is allowed to create a link in a certain directory.

void createLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
                 QString target)

void createLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
                 QString target) = 0
    Create a link.

bool canCreateFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl)

bool canCreateFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl) = 0
    Returns if it is allowed to create files in a certain directory.

```

```
void createFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,  
                QIODevice *content, HandlerTransfer *handlertransfer)
```

```
void createFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,  
                QIODevice *content, HandlerTransfer *handlertransfer) = 0  
    Creates a file with some content.
```

It may use `handlertransfer` for some better integration, like cancel support and progress monitoring.

```
bool canDeleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>  
                   eurl)
```

```
bool canDeleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>  
                   eurl) = 0  
    Returns if it is allowed to delete a certain file/directory/link/...
```

```
void deleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl)
```

```
void deleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl) =  
    0  
    Delete a file/directory/link/...
```

```
bool canRenameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>  
                   src)
```

```
bool canRenameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>  
                   src) = 0  
    Returns if it is allowed to rename a certain file, directory, link, ... (see renameItem).
```

```
void renameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> src,  
                QString destpath)
```

```
void renameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> src,  
                QString destpath) = 0  
    Renames an item to another path.
```

It can be a simple filename change or also changes in the deeper path segments.

```
QList<std::shared_ptr<sh::actions::AbstractActionItem>> getActions (const  
                                                                    QList<std::shared_ptr<const  
                                                                    sh::filesystem::Eurl>> eurls)
```

```
QList<std::shared_ptr<sh::actions::AbstractActionItem>> getActions (const  
                                                                    QList<std::shared_ptr<const  
                                                                    sh::filesystem::Eurl>> eurls)  
                                                                    = 0
```

Returns a list of actions (see former example) for showing for certain files.

```
bool requiresResolvedLinks ()
```

Returns if this handler requires to be used by the engine with resolved links.

```
bool isWellKnownScheme ()
```

Returns if the scheme for this handler is a well known one (which external programs typically would understand).

```
void configureItems (sh::filesystem::Operation *op,    QList<std::shared_ptr<FileSystemNode>>  
                   nodes)
```

Configure newly created nodes (e.g. setting another icon or changing the display name).

```
QList<QString> listExtendedAttributes (sh::filesystem::Operation *op, std::shared_ptr<const  
                                     sh::filesystem::Eurl> eurl)
```

Fetches the list of available extended attributes for an entry.

```
quint64 getExtendedAttributeSize (sh::filesystem::Operation *op,    std::shared_ptr<const  
                                     sh::filesystem::Eurl> eurl, QString attribute)
```

Returns the size of value for one extended attribute for an entry.



`QByteArray` **getExtendedAttribute** (*sh::filesystem::Operation* \*op, `std::shared_ptr<const sh::filesystem::Eurl>` eurl, `QString` attribute)  
Returns the value for one extended attribute for an entry.

void **setExtendedAttribute** (*sh::filesystem::Operation* \*op, `std::shared_ptr<const sh::filesystem::Eurl>` eurl, `QString` attribute, `QByteArray` value)  
Sets the value for one extended attribute for an entry.

void **removeExtendedAttribute** (*sh::filesystem::Operation* \*op, `std::shared_ptr<const sh::filesystem::Eurl>` eurl, `QString` attribute)  
Removes one extended attributes for an entry.

`QMap<QString, QString>` **getCustomAttributes** (*sh::filesystem::Operation* \*op, `std::shared_ptr<const sh::filesystem::Eurl>` eurl)  
Returns the custom attributes for an entry.

In contrast to extended attributes, the custom attributes are not directly stored in the filesystem this way, but handle implementation specific aspects (like permissions).

void **setCustomAttribute** (*sh::filesystem::Operation* \*op, `std::shared_ptr<const sh::filesystem::Eurl>` eurl, `QString` attribute, `QString` value)  
Sets the value for one custom attribute for an entry.

See also `getCustomAttributes`.

void **deleteDirectoryIfEmpty** (*sh::filesystem::Operation* \*op, `std::shared_ptr<const sh::filesystem::Eurl>` eurl)  
Deletes a directory only if it is empty.

void **viewEnteredDirectory** (`std::shared_ptr<const sh::filesystem::Eurl>`)  
Callback for preparing stuff when the view entered some directory (e.g. for enabling watchers). See also `viewLeftDirectory`.

void **viewLeftDirectory** (`std::shared_ptr<const sh::filesystem::Eurl>`)  
Callback for preparing stuff when the view left some directory. See also `viewEnteredDirectory`.

void **search** (*sh::filesystem::Operation* \*op, `std::shared_ptr<const sh::filesystem::Eurl>` eurl, `std::function<void>` std::shared\_ptr<const sh::filesystem::Eurl>, `QList<std::shared_ptr<sh::search::SearchCriterion>>`, *sh::filesystem::FilesystemNodeType* ftype  
> `addfile`, `std::function<void>`std::shared\_ptr<const sh::filesystem::Eurl>> `visitdir`, `QList<std::shared_ptr<sh::search::SearchCriterion>>` criteria)  
Helps searching for files.

Override this method with a behavior identical to the default, if a specialized implementation can be much faster than the default one.

void **customizeUi** (`std::shared_ptr<sh::filesystem::FilesystemNode>` node, `bool` \*showSearchPanel)  
Creates a custom gui, which becomes part of the fileview.

*sh::filesystem::FilesystemModel* \***model** ()  
A convenience shortcut to the *sh::filesystem::FilesystemModel* (a singleton).

void **doShutdown** ()  
Executes singleton shutdown.

void **shutdown** ()  
Shutdown down this singleton.

bool **isAlive** ()  
Returns if this singleton is alive (true until its shutdown begins).

## Public Static Functions

```
QString mountDevice (QString gvolumeuuid, sh::actions::ActionExecutionInfo *info)
QString mountLocation (QString gurl, sh::actions::ActionExecutionInfo *info)
QString unmountLocation (QString gurl, sh::actions::ActionExecutionInfo *info)
QByteArray eurl2gurlb (std::shared_ptr<const sh::filesystem::Eurl> eurl)
QString eurl2gurl (std::shared_ptr<const sh::filesystem::Eurl> eurl)
std::shared_ptr<const sh::filesystem::Eurl> gurl2eurl (QString gurl)
void logIfError (void *error)
void throwIfError (void *error)
```

## Private Functions

```
GnomeIOSmbFilesystemHandler ()
```

### 10.1.195 Class *sh::filesystemhandlers::LocalFilesystemHandler*

```
class sh::filesystemhandlers::LocalFilesystemHandler : public sh::filesystem::FilesystemHandler, public sh::filesystemhandlers::FilesystemHandler
    A filesystem handler for the local filesystem.
```

Subclassed by *sh::filesystemhandlers::WindowsNetworkFilesystemHandler*

## Public Functions

```
void itemList (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
               sh::filesystem::FilesystemNodeType type, sh::filesystem::FilesystemNodeListEditor
               *list) override
void configureItems (sh::filesystem::Operation *op, QList<std::shared_ptr<sh::filesystem::FilesystemNode>>
                    nodes) override
    Configure newly created nodes (e.g. setting another icon or changing the display name).
sh::filesystem::FilesystemNodeType getType (sh::filesystem::Operation *op, std::shared_ptr<const
                                             sh::filesystem::Eurl> eurl) override
qint64 getSize (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl)
               override
QDateTime getMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                   eurl) override
void setMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
              QDateTime mtime) override
QString getMimeType (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl) override
QString getLinkTarget (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                      eurl) override
bool canGetFileContent (sh::filesystem::Operation *op, std::shared_ptr<const
                   sh::filesystem::Eurl> eurl) override
std::shared_ptr<QIODevice> getFileContent (sh::filesystem::Operation *op,
                                          std::shared_ptr<const sh::filesystem::Eurl> eurl)
                                          override
```



```

bool canCreateDirectory (sh::filesystem::Operation *op, std::shared_ptr<const
                        sh::filesystem::Eurl> eurl) override
void createDirectory (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                        eurl) override
bool canCreateLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                        eurl) override
void createLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
                  QString target) override
bool canCreateFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                        eurl) override
void createFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
                  QIODevice *content, HandlerTransfer *handlertransfer) override
bool canDeleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                        eurl) override
void deleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl)
override
bool canRenameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                        src) override
void renameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> src,
                  QString destpath) override
QList<std::shared_ptr<sh::actions::AbstractActionItem>> getActions (const
                                                                    QList<std::shared_ptr<const
                                                                    sh::filesystem::Eurl>> eurls)
                                                                    override
QList<QString> listExtendedAttributes (sh::filesystem::Operation *op, std::shared_ptr<const
                                                                    sh::filesystem::Eurl> eurl) override
quint64 getExtendedAttributeSize (sh::filesystem::Operation *op, std::shared_ptr<const
                                                                    sh::filesystem::Eurl> eurl, QString attribute) override
QByteArray getExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
                                                                    sh::filesystem::Eurl> eurl, QString attribute) override
void setExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
                                                                    sh::filesystem::Eurl> eurl, QString attribute,
                                                                    QByteArray value)
                                                                    override
void removeExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
                                                                    sh::filesystem::Eurl> eurl, QString attribute) override
QMap<QString, QString> getCustomAttributes (sh::filesystem::Operation *op,
                                                                    std::shared_ptr<const sh::filesystem::Eurl>
                                                                    eurl) override
void setCustomAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
                                                                    sh::filesystem::Eurl> eurl, QString attribute,
                                                                    QString value) override
bool requiresResolvedLinks () override
    Returns if this handler requires to be used by the engine with resolved links.
bool isWellKnownScheme () override
    Returns if the scheme for this handler is a well known one (which external programs typically would
    understand).
QString eurlToLocal (std::shared_ptr<const sh::filesystem::Eurl> eurl)
std::shared_ptr<const sh::filesystem::Eurl> localToEurl (const QString local)

```

void **itemlist** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, *sh::filesystem::FilesystemNodeType* type, *sh::filesystem::FilesystemNodeListEditor* \*list) = 0

Determine a list of subelements in a certain directory.

*sh::filesystem::FilesystemNodeType* **getType** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0

Determine if a file is regular, or a dir, or a link, ...

qint64 **getSize** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0

Determine the size of a file.

QDateTime **getMtime** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0

Determine the mtime of a file.

void **setMtime** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QDateTime time) = 0

Set the mtime of a file.

QString **getMimeType** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0

Determine mime type of a file.

QString **getLinkTarget** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0

Resolve a link.

QList<QString> **listExtendedAttributes** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)

Fetches the list of available extended attributes for an entry.

quint64 **getExtendedAttributeSize** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QString attribute)

Returns the size of value for one extended attribute for an entry.

QByteArray **getExtendedAttribute** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QString attribute)

Returns the value for one extended attribute for an entry.

void **setExtendedAttribute** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QString attribute, QByteArray value)

Sets the value for one extended attribute for an entry.

void **removeExtendedAttribute** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QString attribute)

Removes one extended attributes for an entry.

QMap<QString, QString> **getCustomAttributes** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)

Returns the custom attributes for an entry.

In contrast to extended attributes, the custom attributes are not directly stored in the filesystem this way, but handle implementation specific aspects (like permissions).

void **setCustomAttribute** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QString attribute, QString value)

Sets the value for one custom attribute for an entry.

See also getCustomAttributes.

bool **canGetFileContent** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0

Returns if it is allowed to get the content of a certain file.

```

std::shared_ptr<QIODevice> getFileContent (sh::filesystem::Operation *op,
                                           std::shared_ptr<const sh::filesystem::Eurl> eurl) =
                                           0
    Get the content of a file.

bool canCreateDirectory (sh::filesystem::Operation *op, std::shared_ptr<const
                        sh::filesystem::Eurl> eurl) = 0
    Returns if it is allowed to create subdirectories in a certain directory.

void createDirectory (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                     eurl) = 0
    Create a directory.

bool canCreateLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                   eurl) = 0
    Returns if it is allowed to create a link in a certain directory.

void createLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
                 QString target) = 0
    Create a link.

bool canCreateFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                   eurl) = 0
    Returns if it is allowed to create files in a certain directory.

void createFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
                 QIODevice *content, HandlerTransfer *handlertransfer) = 0
    Creates a file with some content.

    It may use handlertransfer for some better integration, like cancel support and progress monitoring.

bool canDeleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                   eurl) = 0
    Returns if it is allowed to delete a certain file/directory/link/...

void deleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl) =
    0
    Delete a file/directory/link/...

void deleteDirectoryIfEmpty (sh::filesystem::Operation *op, std::shared_ptr<const
                             sh::filesystem::Eurl> eurl)
    Deletes a directory only if it is empty.

bool canRenameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                   src) = 0
    Returns if it is allowed to rename a certain file, directory, link, ... (see renameItem).

void renameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> src,
                 QString destpath) = 0
    Renames an item to another path.

    It can be a simple filename change or also changes in the deeper path segments.

QList<std::shared_ptr<sh::actions::AbstractActionItem>> getActions (const
                                                                QList<std::shared_ptr<const
                                                                sh::filesystem::Eurl>> eurls)
                                                                = 0
    Returns a list of actions (see former example) for showing for certain files.

void viewEnteredDirectory (std::shared_ptr<const sh::filesystem::Eurl>)
    Callback for preparing stuff when the view entered some directory (e.g. for enabling watchers). See also
    viewLeftDirectory.

void viewLeftDirectory (std::shared_ptr<const sh::filesystem::Eurl>)
    Callback for preparing stuff when the view left some directory. See also viewEnteredDirectory.

```

```
void search (sh::filesystem::Operation *op, std::shared_ptr<const  
            sh::filesystem::Eurl> eurl, std::function<void> std::shared_ptr<const  
            sh::filesystem::Eurl>, QList<std::shared_ptr<sh::search::SearchCriterion>>,  
            sh::filesystem::FilesystemNodeType ftype  
> addfile, std::function<void>std::shared_ptr<const sh::filesystem::Eurl>> visitdir,  
            QList<std::shared_ptr<sh::search::SearchCriterion>> criteria)Helps searching for files.
```

Override this method with a behavior identical to the default, if a specialized implementation can be much faster than the default one.

```
void customizeUi (std::shared_ptr<sh::filesystem::FilesystemNode> node, bool *showSearchPanel)  
Creates a custom gui, which becomes part of the fileview.
```

```
sh::filesystem::FilesystemModel *model ()  
A convenience shortcut to the sh::filesystem::FilesystemModel (a singleton).
```

```
void doShutdown ()  
Executes singleton shutdown.
```

```
void shutdown ()  
Shutdown down this singleton.
```

```
bool isAlive ()  
Returns if this singleton is alive (true until its shutdown begins).
```

### 10.1.196 Class sh::filesystemhandlers::SharccFilesystemHandler

```
class sh::filesystemhandlers::SharccFilesystemHandler : public sh::filesystem::FilesystemHandler, public s  
A filesystem handler for the sharcc archives.
```

This is a brutally easy and inefficient archive format which exists mostly for testing.

#### Public Functions

```
void itemlist (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,  
              sh::filesystem::FilesystemNodeType type, sh::filesystem::FilesystemNodeListEditor  
              *list) override
```

```
void configureItems (sh::filesystem::Operation *op, QList<std::shared_ptr<sh::filesystem::FilesystemNode>>  
                    nodes) override
```

```
sh::filesystem::FilesystemNodeType getType (sh::filesystem::Operation *op, std::shared_ptr<const  
                                             sh::filesystem::Eurl> eurl) override
```

```
qint64 getSize (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl)  
          override
```

```
QDateTime getMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>  
                  eurl) override
```

```
void setMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,  
              QDateTime time) override
```

```
QString getMimeType (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>  
                   eurl) override
```

```
QString getLinkTarget (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>  
                     eurl) override
```

```
bool canGetFileContent (sh::filesystem::Operation *op, std::shared_ptr<const  
                      sh::filesystem::Eurl> eurl) override
```

```

std::shared_ptr<QIODevice> getFileContent (sh::filesystem::Operation *op,
                                           std::shared_ptr<const sh::filesystem::Eurl> eurl)
                                           override

bool canCreateDirectory (sh::filesystem::Operation *op, std::shared_ptr<const
                        sh::filesystem::Eurl> eurl) override

void createDirectory (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl) override

bool canCreateLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                   eurl) override

void createLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
                QString target) override

bool canCreateFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                   eurl) override

void createFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
                QIODevice *content, HandlerTransfer *handlertransfer) override

bool canDeleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                   eurl) override

void deleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl)
                override

bool canRenameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                   src) override

void renameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> src,
                QString destpath) override

QList<std::shared_ptr<sh::actions::AbstractActionItem>> getActions (const
                                                                QList<std::shared_ptr<const
                                                                sh::filesystem::Eurl>> eurls)
                                                                override

void itemlist (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
               sh::filesystem::FilesystemNodeType type, sh::filesystem::FilesystemNodeListEditor
               *list) = 0
    Determine a list of subelements in a certain directory.

void configureItems (sh::filesystem::Operation *op, QList<std::shared_ptr<FilesystemNode>>
                    nodes)
    Configure newly created nodes (e.g. setting another icon or changing the display name).

sh::filesystem::FilesystemNodeType getType (sh::filesystem::Operation *op, std::shared_ptr<const
                                             sh::filesystem::Eurl> eurl) = 0
    Determine if a file is regular, or a dir, or a link, ...

qint64 getSize (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl) = 0
    Determine the size of a file.

QDateTime getMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                   eurl) = 0
    Determine the mtime of a file.

void setMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
               QDateTime time) = 0
    Set the mtime of a file.

QString getMimeType (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl) = 0
    Determine mime type of a file.

```

QString **getLinkTarget** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*>  
eurl) = 0

Resolve a link.

QList<QString> **listExtendedAttributes** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl*> eurl)

Fetches the list of available extended attributes for an entry.

quint64 **getExtendedAttributeSize** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl*> eurl, QString attribute)

Returns the size of value for one extended attribute for an entry.

QByteArray **getExtendedAttribute** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl*> eurl, QString attribute)

Returns the value for one extended attribute for an entry.

void **setExtendedAttribute** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl*> eurl, QString attribute, QByteArray value)

Sets the value for one extended attribute for an entry.

void **removeExtendedAttribute** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl*> eurl, QString attribute)

Removes one extended attributes for an entry.

QMap<QString, QString> **getCustomAttributes** (*sh::filesystem::Operation* \*op,  
std::shared\_ptr<const *sh::filesystem::Eurl*>  
eurl)

Returns the custom attributes for an entry.

In contrast to extended attributes, the custom attributes are not directly stored in the filesystem this way, but handle implementation specific aspects (like permissions).

void **setCustomAttribute** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl*> eurl, QString attribute, QString value)

Sets the value for one custom attribute for an entry.

See also getCustomAttributes.

bool **canGetFileContent** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl*> eurl) = 0

Returns if it is allowed to get the content of a certain file.

std::shared\_ptr<QIODevice> **getFileContent** (*sh::filesystem::Operation* \*op,  
std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) =  
0

Get the content of a file.

bool **canCreateDirectory** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl*> eurl) = 0

Returns if it is allowed to create subdirectories in a certain directory.

void **createDirectory** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*>  
eurl) = 0

Create a directory.

bool **canCreateLink** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*>  
eurl) = 0

Returns if it is allowed to create a link in a certain directory.

void **createLink** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl,  
QString target) = 0

Create a link.



```

bool canCreateFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl) = 0
    Returns if it is allowed to create files in a certain directory.

void createFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
                 QIODevice *content, HandlerTransfer *handlertransfer) = 0
    Creates a file with some content.

    It may use handlertransfer for some better integration, like cancel support and progress monitoring.

bool canDeleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl) = 0
    Returns if it is allowed to delete a certain file/directory/link/...

void deleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl) =
    0
    Delete a file/directory/link/...

void deleteDirectoryIfEmpty (sh::filesystem::Operation *op, std::shared_ptr<const
                             sh::filesystem::Eurl> eurl)
    Deletes a directory only if it is empty.

bool canRenameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                     src) = 0
    Returns if it is allowed to rename a certain file, directory, link, ... (see renameItem).

void renameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> src,
                  QString destpath) = 0
    Renames an item to another path.

    It can be a simple filename change or also changes in the deeper path segments.

QList<std::shared_ptr<sh::actions::AbstractActionItem>> getActions (const
                                                                    QList<std::shared_ptr<const
                                                                    sh::filesystem::Eurl>> eurls)
                                                                    = 0
    Returns a list of actions (see former example) for showing for certain files.

bool requiresResolvedLinks ()
    Returns if this handler requires to be used by the engine with resolved links.

void viewEnteredDirectory (std::shared_ptr<const sh::filesystem::Eurl>)
    Callback for preparing stuff when the view entered some directory (e.g. for enabling watchers). See also
    viewLeftDirectory.

void viewLeftDirectory (std::shared_ptr<const sh::filesystem::Eurl>)
    Callback for preparing stuff when the view left some directory. See also viewEnteredDirectory.

bool isWellKnownScheme ()
    Returns if the scheme for this handler is a well known one (which external programs typically would
    understand).

void search (sh::filesystem::Operation *op, std::shared_ptr<const
            sh::filesystem::Eurl> eurl, std::function<void> std::shared_ptr<const
            sh::filesystem::Eurl>, QList<std::shared_ptr<sh::search::SearchCriterion>>,
            sh::filesystem::FilesystemNodeType ftype
            > addfile, std::function<void> std::shared_ptr<const sh::filesystem::Eurl>> visitdir,
            QList<std::shared_ptr<sh::search::SearchCriterion>> criteria)
    Helps searching for files.

    Override this method with a behavior identical to the default, if a specialized implementation can be much
    faster than the default one.

void customizeUi (std::shared_ptr<sh::filesystem::FilesystemNode> node, bool *showSearchPanel)
    Creates a custom gui, which becomes part of the fileview.

```

*sh::filesystem::FilesystemModel* \***model** ()

A convenience shortcut to the *sh::filesystem::FilesystemModel* (a singleton).

void **doShutdown** ()

Executes singleton shutdown.

void **shutdown** ()

Shutdown down this singleton.

bool **isAlive** ()

Returns if this singleton is alive (true until its shutdown begins).

## Public Static Functions

std::shared\_ptr<const *filesystem::Eurl*> **localToEurl** (const QString *local*)

## Private Functions

**SharcFilesystemHandler** ()

### 10.1.197 Class *sh::filesystemhandlers::SharcFilesystemHandler::ArchivedSizeDetailColumn*

**class** *sh::filesystemhandlers::SharcFilesystemHandler::ArchivedSizeDetailColumn* : **public** *sh::file*

## Public Functions

**ArchivedSizeDetailColumn** ()

QString **name** ()

The internal unique name.

QString **displayName** ()

The display name.

bool **isValueAvailable** (std::shared\_ptr<FilesystemNode> *node*)

Checks if a value for this detail is available for one given node.

QVariant **value** (std::shared\_ptr<FilesystemNode> *node*, bool *ignoreAged* = false)

Returns the value for this detail for one given node.

## Parameters

- *ignoreAged*: If no value should be returned which is older than the last request (not useful for nearly all cases).

QString **displayValue** (std::shared\_ptr<FilesystemNode> *node*, **const** *sh::filesystem::FilesystemModelFileviewProxy* \**viewmodel*)

Returns the stringified value for this details for one given node considering the configuration of a view.

bool **isVisible** ()

If this detail shall be a visible column in the view.

QVariant **requestValue** (std::shared\_ptr<FilesystemNode> *node*, *sh::filesystem::Operation* \**op* = 0, bool *withNodeValues* = false, bool *withOperationsCache* = true)

Requests to determine the value for this detail for one given node. Blocks until the value is available. Get it with the value method afterwards.



bool **sort\_doTypediff** ()

If sorting should separate files and directories.

int **sort\_order** (std::shared\_ptr<FilesystemNode> *n1*, std::shared\_ptr<FilesystemNode> *n2*)

The sorting order.

uint **displayIndex** ()

Controls which place this column should get in the user interface.

QVariant **computeValue** (std::shared\_ptr<FilesystemNode> *node*, *sh::filesystem::Operation* \**op*)

Returns a freshly determined value for this column and the given node. It neither asks nor populates any caches or storages.

int **defaultWidth** ()

bool **isRightAligned** ()

void **applyValueByEurl** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*,  
*sh::filesystem::Operation* \**op*, QVariant *value*)

Applies the detail value to a given eurl (without using any caches).

Used for transferring some details when a file transfer occurs.

Optionally implement the one of the `applyValueBy...` methods and register the instance with *sh::filesystem::FilesystemOperation::addTransferrableDetailColumn*. For performance reasons, you should decide to implement `applyValueByEurl`.

void **applyValueByNode** (std::shared\_ptr<FilesystemNode> *node*, *sh::filesystem::Operation* \**op*,  
QVariant *value*)

Applies the detail value to a given node (without using any caches).

Used for transferring some details when a file transfer occurs.

Optionally implement the one of the `applyValueBy...` methods and register the instance with *sh::filesystem::FilesystemOperation::addTransferrableDetailColumn*. For performance reasons, you should decide to implement `applyValueByEurl`.

## Public Static Functions

QVariant **VALUE\_UNAVAILABLE** ()

A special value expressing that the value is not yet available.

void **registerKnownDetailColumn** (QString *name*, std::shared\_ptr<DetailColumn> *column*)

std::shared\_ptr<DetailColumn> **findKnownDetailColumn** (QString *name*)

## Public Static Attributes

const uint **DISPLAYINDEX\_CORE** = 10000

const uint **DISPLAYINDEX\_INTERESTING** = 20000

const uint **DISPLAYINDEX\_EXOTIC** = 30000

### 10.1.198 Class `sh::filesystemhandlers::SharcFilesystemHandler::QBuffer2`

```
class sh::filesystemhandlers::SharcFilesystemHandler::QBuffer2 : public QBuffer
```

#### Public Functions

```
QBuffer2 (QByteArray *a)
~QBuffer2 ()
```

#### Private Members

```
QByteArray *_a
```

### 10.1.199 Class `sh::filesystemhandlers::WindowsNetworkFilesystemHandler`

```
class sh::filesystemhandlers::WindowsNetworkFilesystemHandler : public sh::filesystemhandlers::LocalFilesystemHandler
```

#### Public Functions

```
void itemList (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
               sh::filesystem::FilesystemNodeType type, sh::filesystem::FilesystemNodeListEditor
               *list) override
```

```
void itemList (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
               sh::filesystem::FilesystemNodeType type, sh::filesystem::FilesystemNodeListEditor
               *list) = 0
```

Determine a list of subelements in a certain directory.

```
void configureItems (sh::filesystem::Operation *op, QList<std::shared_ptr<sh::filesystem::FilesystemNode>>
                    nodes) override
```

Configure newly created nodes (e.g. setting another icon or changing the display name).

```
sh::filesystem::FilesystemNodeType getType (sh::filesystem::Operation *op, std::shared_ptr<const
                                           sh::filesystem::Eurl> eurl) override
```

```
sh::filesystem::FilesystemNodeType getType (sh::filesystem::Operation *op, std::shared_ptr<const
                                           sh::filesystem::Eurl> eurl) = 0
```

Determine if a file is regular, or a dir, or a link, ...

```
qint64 getSize (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl)
               override
```

```
qint64 getSize (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl) = 0
```

Determine the size of a file.

```
QDateTime getMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                   eurl) override
```

```
QDateTime getMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                   eurl) = 0
```

Determine the mtime of a file.

```
void setMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
               QDateTime mtime) override
```

```
void setMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
               QDateTime time) = 0
```

Set the mtime of a file.

```

QString getMimeType (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl) override
QString getMimeType (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl) = 0
    Determine mime type of a file.
QString getLinkTarget (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl) override
QString getLinkTarget (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl) = 0
    Resolve a link.
bool canGetFileContent (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) override
bool canGetFileContent (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) = 0
    Returns if it is allowed to get the content of a certain file.
std::shared_ptr<QIODevice> getFileContent (sh::filesystem::Operation *op,
                    std::shared_ptr<const sh::filesystem::Eurl> eurl)
                    override
std::shared_ptr<QIODevice> getFileContent (sh::filesystem::Operation *op,
                    std::shared_ptr<const sh::filesystem::Eurl> eurl) =
                    0
    Get the content of a file.
bool canCreateDirectory (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) override
bool canCreateDirectory (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) = 0
    Returns if it is allowed to create subdirectories in a certain directory.
void createDirectory (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl) override
void createDirectory (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl) = 0
    Create a directory.
bool canCreateLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl) override
bool canCreateLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl) = 0
    Returns if it is allowed to create a link in a certain directory.
void createLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
                    QString target) override
void createLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
                    QString target) = 0
    Create a link.
bool canCreateFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl) override
bool canCreateFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl) = 0
    Returns if it is allowed to create files in a certain directory.
void createFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
                    QIODevice *content, HandlerTransfer *handlertransfer) override

```

```
void createFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,  
                QIODevice *content, HandlerTransfer *handlertransfer) = 0  
Creates a file with some content.
```

It may use handlertransfer for some better integration, like cancel support and progress monitoring.

```
bool canDeleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>  
                   eurl) override
```

```
bool canDeleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>  
                   eurl) = 0  
Returns if it is allowed to delete a certain file/directory/link/...
```

```
void deleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl)  
               override
```

```
void deleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl) =  
0  
Delete a file/directory/link/...
```

```
bool canRenameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>  
                   src) override
```

```
bool canRenameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>  
                   src) = 0  
Returns if it is allowed to rename a certain file, directory, link, ... (see renameItem).
```

```
void renameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> src,  
                QString destpath) override
```

```
void renameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> src,  
                QString destpath) = 0  
Renames an item to another path.
```

It can be a simple filename change or also changes in the deeper path segments.

```
QList<std::shared_ptr<sh::actions::AbstractActionItem>> getActions (const  
                                                                    QList<std::shared_ptr<const  
                                                                    sh::filesystem::Eurl>> eurls)  
                                                                    override
```

```
QList<std::shared_ptr<sh::actions::AbstractActionItem>> getActions (const  
                                                                    QList<std::shared_ptr<const  
                                                                    sh::filesystem::Eurl>> eurls)  
                                                                    = 0  
Returns a list of actions (see former example) for showing for certain files.
```

```
QList<QString> listExtendedAttributes (sh::filesystem::Operation *op, std::shared_ptr<const  
                                       sh::filesystem::Eurl> eurl) override
```

```
QList<QString> listExtendedAttributes (sh::filesystem::Operation *op, std::shared_ptr<const  
                                       sh::filesystem::Eurl> eurl)  
Fetches the list of available extended attributes for an entry.
```

```
quint64 getExtendedAttributeSize (sh::filesystem::Operation *op, std::shared_ptr<const  
                                   sh::filesystem::Eurl> eurl, QString attribute) override
```

```
quint64 getExtendedAttributeSize (sh::filesystem::Operation *op, std::shared_ptr<const  
                                   sh::filesystem::Eurl> eurl, QString attribute)  
Returns the size of value for one extended attribute for an entry.
```

```
QByteArray getExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const  
                                   sh::filesystem::Eurl> eurl, QString attribute) override
```

```
QByteArray getExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const  
                                   sh::filesystem::Eurl> eurl, QString attribute)  
Returns the value for one extended attribute for an entry.
```

```

void setExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
                        sh::filesystem::Eurl> eurl, QString attribute, QByteArray value)
                        override

void setExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
                        sh::filesystem::Eurl> eurl, QString attribute, QByteArray value)
    Sets the value for one extended attribute for an entry.

void removeExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
                        sh::filesystem::Eurl> eurl, QString attribute) override

void removeExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
                        sh::filesystem::Eurl> eurl, QString attribute)
    Removes one extended attributes for an entry.

QMap<QString, QString> getCustomAttributes (sh::filesystem::Operation *op,
                        std::shared_ptr<const sh::filesystem::Eurl>
                        eurl) override

QMap<QString, QString> getCustomAttributes (sh::filesystem::Operation *op,
                        std::shared_ptr<const sh::filesystem::Eurl>
                        eurl)

    Returns the custom attributes for an entry.

    In contrast to extended attributes, the custom attributes are not directly stored in the filesystem this way,
    but handle implementation specific aspects (like permissions).

void setCustomAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
                        sh::filesystem::Eurl> eurl, QString attribute, QString value) override

void setCustomAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
                        sh::filesystem::Eurl> eurl, QString attribute, QString value)
    Sets the value for one custom attribute for an entry.

    See also getCustomAttributes.

bool requiresResolvedLinks () override
    Returns if this handler requires to be used by the engine with resolved links.

bool isWellKnownScheme () override
    Returns if the scheme for this handler is a well known one (which external programs typically would
    understand).

QString eurlToLocal (std::shared_ptr<const sh::filesystem::Eurl> eurl)

std::shared_ptr<const sh::filesystem::Eurl> localToEurl (const QString local)

void deleteDirectoryIfEmpty (sh::filesystem::Operation *op, std::shared_ptr<const
                        sh::filesystem::Eurl> eurl)
    Deletes a directory only if it is empty.

void viewEnteredDirectory (std::shared_ptr<const sh::filesystem::Eurl>)
    Callback for preparing stuff when the view entered some directory (e.g. for enabling watchers). See also
    viewLeftDirectory.

void viewLeftDirectory (std::shared_ptr<const sh::filesystem::Eurl>)
    Callback for preparing stuff when the view left some directory. See also viewEnteredDirectory.

void search (sh::filesystem::Operation *op, std::shared_ptr<const
                        sh::filesystem::Eurl> eurl, std::function<void> std::shared_ptr<const
                        sh::filesystem::Eurl>, QList<std::shared_ptr<sh::search::SearchCriterion>>,
                        sh::filesystem::FilesystemNodeType ftype
    > addfile, std::function<void>std::shared_ptr<const sh::filesystem::Eurl>> visitdir,
    QList<std::shared_ptr<sh::search::SearchCriterion>> criteria) helps searching for files.

```

Override this method with a behavior identical to the default, if a specialized implementation can be much faster than the default one.

void **customizeUi** (std::shared\_ptr<sh::filesystem::FilesystemNode> node, bool \*showSearchPanel)  
Creates a custom gui, which becomes part of the fileview.

sh::filesystem::FilesystemModel \***model** ()  
A convenience shortcut to the sh::filesystem::FilesystemModel (a singleton).

void **doShutdown** ()  
Executes singleton shutdown.

void **shutdown** ()  
Shutdown down this singleton.

bool **isAlive** ()  
Returns if this singleton is alive (true until its shutdown begins).

### 10.1.200 Class sh::paneldetails::CommonPanelDetails

**class** sh::paneldetails::CommonPanelDetails  
Implementations of common details provider for the details panel.

#### Public Static Functions

void **doInitialize** ()

void **doShutdown** ()

void **\_ViaDetailColumn** (std::shared\_ptr<PanelDetail> detail, std::shared\_ptr<sh::filesystem::FilesystemNode> node, std::shared\_ptr<sh::filesystem::DetailColumn> column, QString key-name, std::function<QString> QVariant  
> valtostring

void **NumberOfSelectedItems** (std::shared\_ptr<PanelDetail>, QList<std::shared\_ptr<sh::filesystem::FilesystemNode>>, sh::filesystem::Operation \*op)

void **TotalFileSize** (std::shared\_ptr<PanelDetail>, QList<std::shared\_ptr<sh::filesystem::FilesystemNode>>, sh::filesystem::Operation \*op)

void **FileType** (std::shared\_ptr<PanelDetail> detail, std::shared\_ptr<sh::filesystem::FilesystemNode> node, sh::filesystem::Operation \*op)

void **FileSize** (std::shared\_ptr<PanelDetail> detail, std::shared\_ptr<sh::filesystem::FilesystemNode> node, sh::filesystem::Operation \*op)

void **FileModificationTime** (std::shared\_ptr<PanelDetail> detail, std::shared\_ptr<sh::filesystem::FilesystemNode> node, sh::filesystem::Operation \*op)

### 10.1.201 Class `sh::paneldetails::PanelDetail`

**class** `sh::paneldetails::PanelDetail` : **public** `QObject`

A detail panel entry.

Is is essentially a list of `PanelDetailRow` and some layout information.

#### Public Functions

**PanelDetail** (int *positionIndex*, int *valueWidthHint*)

Constructed only by the infrastructure and made available otherwise.

`QList<PanelDetailRow>` **rows** ()

Returns the list of detail rows stored in this panel detail.

int **positionIndex** ()

Returns a position index.

It controls the position of this panel detail in relation to the other ones.

int **valueWidthHint** ()

Returns the specified width for the panel detail.

void **setRows** (`QList<PanelDetailRow>` *rows*)

Sets the list of detail rows.

void **addRow** (`PanelDetailRow` *row*)

Adds a new detail row.

void **addOrReplaceRow** (`PanelDetailRow` *row*)

Adds a new detail row (replacing an old one with the same key).

void **\_emit\_linkTriggered** (`QString` *link*)

#### Signals

void **changed** ()

Emitted when data has changed.

void **linkTriggered** (`QString` *link*)

Emitted when a link is triggered by the user.

#### Private Members

`QMutex` **\_rowsmutex**

`QList<PanelDetailRow>` **\_rows**

int **\_positionIndex**

int **\_valueWidthHint**



### 10.1.202 Class `sh::paneldetails::PanelDetailFactory`

**class** `sh::paneldetails::PanelDetailFactory`

Abstract factory for creating panel details for the selected nodes.

Subclassed by `sh::paneldetails::PanelDetailFactoryForFunctionMulti`, `sh::paneldetails::PanelDetailFactoryForFunctionSingle`

#### Public Functions

`std::shared_ptr<PanelDetail> construct (QList<std::shared_ptr<sh::filesystem::FileSystemNode>>  
nodes, sh::filesystem::Operation *op) = 0`

### 10.1.203 Class `sh::paneldetails::PanelDetailFactoryForFunctionMulti`

**class** `sh::paneldetails::PanelDetailFactoryForFunctionMulti : public sh::paneldetails::PanelDetailFactory`

A factory for creating panel details for more than one selected node.

Subclassed by `sh::scripting::api::ApiPanelDetailFactoryMulti`

#### Public Functions

**PanelDetailFactoryForFunctionMulti** (`std::function<void>` `std::shared_ptr<PanelDetail>`,  
`QList<std::shared_ptr<sh::filesystem::FileSystemNode>>`,  
`sh::filesystem::Operation *op`  
> `fcnsetvalue`, `std::function<void>` `QList<std::shared_ptr<sh::filesystem::FileSystemNode>>`, `QString link` >  
`fcnlinktriggered`, `int position`, `int valueWidthHint` Constructed only indirectly.  
`std::shared_ptr<PanelDetail> construct (QList<std::shared_ptr<sh::filesystem::FileSystemNode>>  
nodes, sh::filesystem::Operation *op)`

### 10.1.204 Class `sh::paneldetails::PanelDetailFactoryForFunctionSingle`

**class** `sh::paneldetails::PanelDetailFactoryForFunctionSingle : public sh::paneldetails::PanelDetailFactory`

A factory for creating panel details for just one selected node.

Subclassed by `sh::scripting::api::ApiPanelDetailFactorySingle`

#### Public Functions

**PanelDetailFactoryForFunctionSingle** (`std::function<void>` `std::shared_ptr<PanelDetail>`,  
`std::shared_ptr<sh::filesystem::FileSystemNode>`,  
`sh::filesystem::Operation *op`  
> `fcnsetvalue`, `std::function<void>` `std::shared_ptr<sh::filesystem::FileSystemNode>`, `QString link` >  
`fcnlinktriggered`, `int position`, `int valueWidthHint` Constructed only indirectly.  
`std::shared_ptr<PanelDetail> construct (QList<std::shared_ptr<sh::filesystem::FileSystemNode>>  
nodes, sh::filesystem::Operation *op)`





### Private Members

`QList<std::shared_ptr<PanelDetailFactory>> _factoriesingle`  
`QList<std::shared_ptr<PanelDetailFactory>> _factoriesmulti`  
`QMutex _mutex`

## 10.1.206 Class `sh::paneldetails::PanelDetailRow`

**class** `sh::paneldetails::PanelDetailRow`  
One pair of label text and value text in a panel detail.

### Public Functions

**PanelDetailRow** (`QString key`, `QList<PanelDetailRowValueElement> value`)  
Is intended to be directly constructed from everywhere.

**PanelDetailRow** (`QList<PanelDetailRowValueElement> value`)  
Is intended to be directly constructed from everywhere.

`QString key` ()

`QList<PanelDetailRowValueElement> value` ()

### Private Members

`QString _key`

`QList<PanelDetailRowValueElement> _value`

## 10.1.207 Class `sh::paneldetails::PanelDetailRowValueElement`

**class** `sh::paneldetails::PanelDetailRowValueElement`  
One element of a detail row's value.

This can be something like a string, an icon or a link button.

### Public Functions

**PanelDetailRowValueElement** ()  
A placeholder for a value which is not yet arrived. It could be visualized with a loading animation or '...'.  
Is intended to be directly constructed from everywhere.

**PanelDetailRowValueElement** (`QString text`)  
A piece of text. Is intended to be directly constructed from everywhere.

**PanelDetailRowValueElement** (`QIcon icon`)  
An icon. Is intended to be directly constructed from everywhere.

**PanelDetailRowValueElement** (`QString text`, `QString linktarget`, `bool autohide`)  
A link button. Is intended to be directly constructed from everywhere.

`QString text` ()  
Returns the text.

QIcon **icon** ()  
Returns the icon.

QString **linktarget** ()  
Returns the link target.

bool **linkautohide** ()  
Returns if this link shall automatically hide (e.g. when the mouse disappears).

bool **isWaiting** ()  
Returns if it is currently waiting for an update.

### Private Members

QString **\_text**

QIcon **\_icon**

QString **\_linktarget**

bool **\_linkautohide** = false

bool **\_iswaiting** = false

## 10.1.208 Class sh::scripting::Api

**class** *sh::scripting::Api*  
Interoperation layer between the Shallot core and a plugin interpreter.

### Public Functions

**Api** ()

**~Api** ()

QStringList **classes** ()

QStringList **rootMembers** ()

*ApiClass* \***getClass** (QString *name*)

*ApiClass* \***getClassByIdName** (QString *typeidName*)

void \***rootMember** (QString *name*)

*ApiClass* \***rootMemberClass** (QString *name*)

### Private Functions

void **registerClass** (QString *name*, *ApiClass* \*\**apiclass*, **const** std::type\_info &*nativeType*)

void **registerMethod** (*ApiClass* \**apiclass*, QString *name*)

void **registerRootObjectMember** (QString *name*, *ApiClass* \**cls*, void \**member*)

### Private Members

```
QHash<QString, ApiClass*> _classes  
QHash<QString, ApiClass*> _classesByNativeType  
QHash<QString, void*> _rootMembers  
QHash<QString, ApiClass*> _rootMembersClasses
```

## 10.1.209 Class `sh::scripting::ApiClass`

**class** `sh::scripting::ApiClass`  
Data structure for one api-aware core class.

### Public Functions

```
ApiClass (QString name)  
    Constructed only indirectly.  
  
~ApiClass ()  
  
QString name ()  
  
QStringList methods ()  
  
void addMethod (ApiMethod *m)  
  
ApiMethod *getMethod (QString name)
```

### Private Members

```
QString _name  
QHash<QString, ApiMethod*> _methods
```

## 10.1.210 Class `sh::scripting::ApiMethod`

**class** `sh::scripting::ApiMethod`  
Data structure for one api-aware core method.

### Public Functions

```
ApiMethod (QString name)  
    Constructed only indirectly.  
  
QString name ()
```

### 10.1.211 Class `sh::scripting::PythonScriptInterpreter`

**class** `sh::scripting::PythonScriptInterpreter` : **public** `sh::scripting::ScriptInterpreter`, **public** `sh::base::Single`  
 Integration of the Python language for scripting.

Find also the scripting manual for all details about the scripting interface.

#### Public Functions

```
~PythonScriptInterpreter ()
bool isAvailable ()
void initializeInterpreter ()
QString filesuffix ()
void execute (QString script)
void doInitialize ()
    Executes singleton initialization.
void shutdown ()
    Shutdown down this singleton.
bool isAlive ()
    Returns if this singleton is alive (true until its shutdown begins).
```

#### Public Members

```
PyObject * _Exception
PyObject * _CancelException
PyObject * _ThreadAbortException
```

#### Public Static Functions

```
void registerMethod (sh::scripting::ApiClass *class, QString name, PyCFunction fcnptr)
void throwNativeExceptionFromPythonTraceback ()
```

#### Public Static Attributes

```
QHash<void*, PyObject*> _buddies
QHash<void*, std::weak_ptr<void>> _natives
```

### Private Functions

**PythonScriptInterpreter()**

### Private Static Functions

**PyObject \*getApiModuleDef()**

### Private Static Attributes

**void \*apiModuleDef**  
*sh::scripting::Api \*api* = new *sh::scripting::Api()*  
**QHash<QString, PyCFunction> \_methods**  
**PyObject \*modtraceback**

## 10.1.212 Class *sh::scripting::ScriptInterpreter*

**class** *sh::scripting::ScriptInterpreter*  
The base class for integration of a script language.  
Subclassed by *sh::scripting::PythonScriptInterpreter*

### Public Functions

**bool isAvailable()**  
**void initializeInterpreter()**  
**QString filesuffix()**  
**void execute**(**QString**)  
**~ScriptInterpreter()**

## 10.1.213 Class *sh::scripting::ScriptingEngine*

**class** *sh::scripting::ScriptingEngine* : **public** **QObject**, **public** *sh::base::Singleton*  
The Scripting engine loads scripts by means of the registered interpreters.

### Public Functions

**void loadScript**(**QString** *script*)  
Loads a plugin script from file.  
**void doShutdown()**  
Executes singleton shutdown.  
**void shutdown()**  
Shutdown down this singleton.  
**bool isAlive()**  
Returns if this singleton is alive (true until its shutdown begins).

## Private Functions

**ScriptingEngine** ()

void **registerInterpreter** (std::shared\_ptr<sh::scripting::ScriptInterpreter> interpreter)

## Private Members

QMap<QString, std::shared\_ptr<sh::scripting::ScriptInterpreter>> **\_interpreters**

## Private Static Attributes

std::shared\_ptr<sh::configuration::ConfigurationValue> **cfgvallastStartupFailed** = sh::configuration::ConfigurationMa

**class ActionPluginLoadCrashedAgain** : public sh::actions::ActionActionItem

## Public Functions

**ActionPluginLoadCrashedAgain** (QString path, QMutex \*mutex)

void **execute** ()

Executes this action.

void **execute** (sh::actions::ActionExecutionInfo \*info)

Executes this action.

QKeySequence **shortcuthint** ()

Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()

Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence shortcut, bool triggersOnCurrentDirectory = false)

Sets the keyboard shortcut for triggering this action.

QString **text** ()

Returns the displayed text for this action.

QString **icon** ()

Returns the icon for this action.

bool **enabled** ()

Checks if this action is enabled.

bool **isChecked** ()

Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () const

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

```
void setText (QString text)  
    Sets the displayed text.  
  
void setIcon (QString icon)  
    Sets the icon.  
  
void setEnabled (bool enabled)  
    Sets if the item is enabled.  
  
void setChecked (bool checked)  
    Sets if the item is checked (has a cross in the ui).  
  
void setVisible (bool visible)  
    Sets the visibility of this item.  
  
bool visible ()  
    Checks the visibility of this item (non-recursively).  
  
std::weak_ptr<AbstractActionItem> parentAction ()  
    Returns the parent action, if it is added to a container.  
  
void _setParentAction (std::shared_ptr<AbstractActionItem> parent)  
    Sets the parent action. .  
  
void initializeAsync (std::function<void>  
    > oninitialized = 0) Asynchronously initializes the action.  
  
void initializeSync ()  
    Synchronously initializes the action.  
  
bool isInitialized ()  
    Checks if this action is initialized.  
  
bool isInitializing ()  
    Checks if this action is initializing.
```

## Signals

```
void changed ()  
    Emits when some data changed.
```

```
class ScriptedException: public sh::exceptions::Exception  
    Exception for errors within the scripting engine and interpreters.
```

## Public Functions

```
ScriptedException (sh::exceptions::ExceptionData data)  
  
QString message () const  
  
QString name () const  
  
QString classes () const  
  
QString details () const  
  
QString callstack () const  
  
QString auxiliary () const  
  
QString customValue (QString key) const  
  
bool isRuntimeException () const
```



```

bool isProgramException() const
bool isRetryable() const
bool isResumeable() const
bool isDetailsAreInteresting() const
int autoRetryRecommendedIn() const
void setResumeable(bool v)
void setReTryable(bool v)
void setCustomValue(QString key, QString value)
bool isClass(QString classname)
sh::exceptions::ExceptionData data()

```

### Public Static Functions

```

template<class Handler>
std::shared_ptr<RegisterHandler<Handler>> createRegisterHandler(Handler handler,
                                                                QStack<Handler>
                                                                *stack)

void executeGuarded(std::function<void>
    > fct int flags = 0) Executes some code with some standard exection handling around.

Parameters
    • fct: The code to execute guarded.
    • flags: Flags of ExecuteGuardFlag for choosing the behavior.

void executeGuarded_errorpanel(std::function<void>
    > fct int flags = 0

QString _demangle(QString l)

void exceptionDialog(sh::exceptions::Exception &e, bool *doRetry)

```

### Public Static Attributes

```

const QString UNDEFINED_ERROR_OCCURRED = QObject::tr("An unspecified error occurred.")
const QString SHALLOT_MUST_CLOSE_TEXT = QObject::tr("Your Shallot process is disturbed by an error and needs to be restarted.")
const QString Value_isShallotException = "_isShallotException"

```

## 10.1.214 Class sh::scripting::ScriptingEngine::ActionPluginLoadCrashedAgain

```

class sh::scripting::ScriptingEngine::ActionPluginLoadCrashedAgain : public sh::actions::ActionAction

```

## Public Functions

**ActionPluginLoadCrashedAgain** (QString *path*, QMutex \**mutex*)

void **execute** ()

Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)

Executes this action.

QKeySequence **shortcuthint** ()

Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()

Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)

Sets the keyboard shortcut for triggering this action.

QString **text** ()

Returns the displayed text for this action.

QString **icon** ()

Returns the icon for this action.

bool **enabled** ()

Checks if this action is enabled.

bool **isChecked** ()

Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<AbstractActionItem> **parentAction** ()

Returns the parent action, if it is added to a container.

```

void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.

```

## Signals

```

void changed ()
    Emits when some data changed.

```

### 10.1.215 Class `sh::scripting::ScriptingEngine::ScriptedException`

```

class sh::scripting::ScriptingEngine::ScriptedException : public sh::exceptions::Exception
    Exception for errors within the scripting engine and interpreters.

```

#### Public Functions

```

ScriptedException (sh::exceptions::ExceptionData data)

QString message () const

QString name () const

QString classes () const

QString details () const

QString callstack () const

QString auxiliary () const

QString customValue (QString key) const

bool isRuntimeException () const

bool isProgramException () const

bool isRetryable () const

bool isResumeable () const

bool isDetailsAreInteresting () const

int autoRetryRecommendedIn () const

void setResumeable (bool v)

void setReTryable (bool v)

void setCustomValue (QString key, QString value)

bool isClass (QString classname)

```

*sh::exceptions::ExceptionData* **data** ()

### Public Static Functions

template<class **Handler**>  
std::shared\_ptr<RegisterHandler<*Handler*>> **createRegisterHandler** (*Handler* handler, QS-  
tack<*Handler*> \*stack)

void **executeGuarded** (std::function<void>  
> *fct* int *flags* = 0 Executes some code with some standard execution handling around.

#### Parameters

- *fct*: The code to execute guarded.
- *flags*: Flags of *ExecuteGuardFlag* for choosing the behavior.

void **executeGuarded\_errorpanel** (std::function<void>  
> *fct* int *flags* = 0

QString **\_demangle** (QString *l*)

void **exceptionDialog** (*sh::exceptions::Exception* &*e*, bool \**doRetry*)

### Public Static Attributes

const QString **UNDEFINED\_ERROR\_OCCURRED** = QObject::tr("An unspecified error occurred.")

const QString **SHALLOT\_MUST\_CLOSE\_TEXT** = QObject::tr("Your Shallot process is disturbed by an error and needs to be c

const QString **Value\_isShallotException** = "\_isShallotException"

## 10.1.216 Class *sh::scripting::api::ApiActionActionItem*

class *sh::scripting::api::ApiActionActionItem* : public *sh::actions::ActionActionItem*

### Public Functions

**ApiActionActionItem** (QString *text*, bool *enabled*, QString *icon*, int *defaultActionPrecedence*)

void **\_execute** ()

void **\_initializeSync** ()

void **action** (*sh::actions::ActionExecutionInfo* \**info*) **override**  
The action implementation, i.e. what the actions should actually do.

void **initialize** () **override**  
Initialize the action. This should make the time-consuming parts, e.g. for determining a label or enabled state.

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)  
Executes this action.

`QKeySequence` **shortcutHint** ()  
Returns the keyboard shortcut for triggering this action.

`bool` **shortcutHintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

`void` **setShortcutHint** (`QKeySequence` *shortcut*, `bool` *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

`QString` **text** ()  
Returns the displayed text for this action.

`QString` **icon** ()  
Returns the icon for this action.

`bool` **enabled** ()  
Checks if this action is enabled.

`bool` **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

`bool` **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

`int` **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

`void` **setText** (`QString` *text*)  
Sets the displayed text.

`void` **setIcon** (`QString` *icon*)  
Sets the icon.

`void` **setEnabled** (`bool` *enabled*)  
Sets if the item is enabled.

`void` **setChecked** (`bool` *checked*)  
Sets if the item is checked (has a cross in the ui).

`void` **setVisible** (`bool` *visible*)  
Sets the visibility of this item.

`bool` **visible** ()  
Checks the visibility of this item (non-recursively).

`std::weak_ptr<AbstractActionItem>` **parentAction** ()  
Returns the parent action, if it is added to a container.

`void` **\_setParentAction** (`std::shared_ptr<AbstractActionItem>` *parent*)  
Sets the parent action. .

`void` **initializeAsync** (`std::function<void>`  
> *oninitialized* = 0) Asynchronously initializes the action.

`void` **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

### Public Members

std::function<void ()> **\_initialize**  
std::function<void (*sh::actions::ActionExecutionInfo*\*)> **\_action**

### Signals

void **changed** ()  
Emits when some data changed.

## 10.1.217 Class *sh::scripting::api::ApiActionFactory*

```
class sh::scripting::api::ApiActionFactory : public sh::actions::AbstractActionFactory
```

### Public Functions

**ApiActionFactory** (QString *category*, QList<std::shared\_ptr<*actions::Predicate*>> *predicates*)  
std::shared\_ptr<*sh::actions::AbstractActionItem*> **construct** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>>  
nodes)  
std::shared\_ptr<ActionInstantiation> **actionAvailable** (ActionInstantiation \**instantiation*)

### Public Members

std::function< std::shared\_ptr< *sh::actions::AbstractActionItem* >QList< std::shared\_ptr<

## 10.1.218 Class *sh::scripting::api::ApiDetailColumn*

```
class sh::scripting::api::ApiDetailColumn : public sh::filesystem::DetailColumn
```

### Public Functions

**ApiDetailColumn** (QString *name*, QString *displayName*, int *displayIndex*, bool *sort\_doTypediff*, int  
defaultWidth, bool *isRightAligned*)  
QVariant **determineValue** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> node,  
*sh::filesystem::Operation* \**op*)  
void **applyValue** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*, *sh::filesystem::Operation* \**op*,  
QVariant *value*)  
QString **name** ()  
The internal unique name.  
QString **displayName** ()  
The display name.

bool **isValueAvailable** (std::shared\_ptr<FilesystemNode> *node*)

Checks if a value for this detail is available for one given node.

QVariant **value** (std::shared\_ptr<FilesystemNode> *node*, bool *ignoreAged* = false)

Returns the value for this detail for one given node.

#### Parameters

- *ignoreAged*: If no value should be returned which is older than the last request (not useful for nearly all cases).

QString **displayValue** (std::shared\_ptr<FilesystemNode> *node*, **const** *sh::filesystem::FilesystemModelFileviewProxy \*viewmodel*)

Returns the stringified value for this details for one given node considering the configuration of a view.

bool **isVisible** ()

If this detail shall be a visible column in the view.

QVariant **requestValue** (std::shared\_ptr<FilesystemNode> *node*, *sh::filesystem::Operation \*op* = 0, bool *withNodeValues* = false, bool *withOperationsCache* = true)

Requests to determine the value for this detail for one given node. Blocks until the value is available. Get it with the value method afterwards.

bool **sort\_doTypediff** ()

If sorting should separate files and directories.

int **sort\_order** (std::shared\_ptr<FilesystemNode> *n1*, std::shared\_ptr<FilesystemNode> *n2*)

The sorting order.

uint **displayIndex** ()

Controls which place this column should get in the user interface.

QVariant **computeValue** (std::shared\_ptr<FilesystemNode> *node*, *sh::filesystem::Operation \*op*)

Returns a freshly determined value for this column and the given node. It neither asks nor populates any caches or storages.

int **defaultWidth** ()

bool **isRightAligned** ()

void **applyValueByEurl** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*, *sh::filesystem::Operation \*op*, QVariant *value*)

Applies the detail value to a given eurl (without using any caches).

Used for transferring some details when a file transfer occurs.

Optionally implement the one of the `applyValueBy...` methods and register the instance with `sh::filesystem::FilesystemOperation::addTransferrableDetailColumn`. For performance reasons, you should decide to implement `applyValueByEurl`.

void **applyValueByNode** (std::shared\_ptr<FilesystemNode> *node*, *sh::filesystem::Operation \*op*, QVariant *value*)

Applies the detail value to a given node (without using any caches).

Used for transferring some details when a file transfer occurs.

Optionally implement the one of the `applyValueBy...` methods and register the instance with `sh::filesystem::FilesystemOperation::addTransferrableDetailColumn`. For performance reasons, you should decide to implement `applyValueByEurl`.

## Public Members

```
std::function<QString (std::shared_ptr<sh::filesystem::FilesystemNode>, sh::filesystem::Operation*)>
    _determineValue
std::function<void (std::shared_ptr<sh::filesystem::Eurl>, sh::filesystem::Operation*, QString)>
    _applyValue
```

## Public Static Functions

```
QVariant VALUE_UNAVAILABLE ()
    A special value expressing that the value is not yet available.
void registerKnownDetailColumn (QString name, std::shared_ptr<DetailColumn> column)
std::shared_ptr<DetailColumn> findKnownDetailColumn (QString name)
```

## Public Static Attributes

```
const uint DISPLAYINDEX_CORE = 10000
const uint DISPLAYINDEX_INTERESTING = 20000
const uint DISPLAYINDEX_EXOTIC = 30000
```

### 10.1.219 Class sh::scripting::api::ApiFilePropertyDialogTab

```
class sh::scripting::api::ApiFilePropertyDialogTab : public sh::ui::FilePropertyDialogTab
```

## Public Functions

```
ApiFilePropertyDialogTab (QString title, QList<PropertyConfig> properties)
QString title () override
    The tab title. .
QList<QString> properties () override
    Returns the list of property labels (one entry for each widget).
    Each property is displayed with a header and a widget (created in createWidget) with some content (populated in updateWidget).
void populateWidget (int i, sh::ui::FilePropertyDialogTabActionsView *widget) override
    Populates the tab widget for the i-th property.
    Typically uses the FilePropertyDialog::create* methods to create sub-widgets.
    See also dialog().
void updateWidget (int i, sh::ui::FilePropertyDialogTabActionsView *widget,
    sh::filesystem::Operation *op) override
    Populates the widget for the i-th property with actual content. .
QString titleWithoutMnemonic ()
    The tab title without mnemonic.
```



void **refresh** ()

Refreshes the complete content.

Typically called after some user actions in a property widget require that all the other content must be refreshed.

QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> **nodes** ()

The nodes to show.

FilePropertyDialogTabActionsView \***widgetAt** (int *i*)

Returns the widget for the i-th property.

int **widgetCount** ()

Returns the number of widgets.

std::shared\_ptr<FilePropertyDialog> **dialog** ()

Returns the associated dialog.

## Public Members

std::function< QList< QString >int, sh::filesystem::Operation \*, qintptr> > **\_updateWidg**

std::function<void (QString) > **\_buttonTriggered**

## Public Static Attributes

const int **PROPERTY\_TYPE\_STRING** = 1

const int **PROPERTY\_TYPE\_STRINGMAP** = 2

const int **PROPERTY\_TYPE\_ICONTEXTBANNER** = 3

## Private Types

typedef QPair<QString, QString> **StringPair**

## Private Members

QString **\_title**

QList<*PropertyConfig*> **\_properties**

class **PropertyConfig**

## Public Types

typedef QPair<QString, QString> **ButtonConfig**

### Public Functions

**PropertyConfig** (QString *title*, int *propertytype*, QList<*ButtonConfig*> *buttons*)

### Public Members

QString **title**

int **propertytype**

QList<*ButtonConfig*> **buttons**

## 10.1.220 Class sh::scripting::api::ApiFilePropertyDialogTabFactory

```
class sh::scripting::api::ApiFilePropertyDialogTabFactory : public sh::ui::FilePropertyDialogTabFactory
```

### Public Functions

**ApiFilePropertyDialogTabFactory** ()

std::shared\_ptr<sh::ui::FilePropertyDialogTab> **construct** (std::shared\_ptr<sh::ui::FilePropertyDialog>  
dialog)

std::shared\_ptr<sh::ui::FilePropertyDialogTab> **construct** (std::shared\_ptr<sh::ui::FilePropertyDialog>  
dialog) = 0

Constructs a fresh instance of a *FilePropertyDialogTab* implementation. .

### Public Members

std::function< std::shared\_ptr< sh::scripting::api::ApiFilePropertyDialogTab >> > **\_construct**

### Public Static Attributes

**const** int **REGISTER\_FILEPROPERTYDIALOGTAB\_INDEX\_CORE** = 10000

Base value for display indexes of core (i.e. maximum important) tabs.

Used in *FilePropertyDialog::addTabFactory*.

**const** int **REGISTER\_FILEPROPERTYDIALOGTAB\_INDEX\_VERYIMPORTANT** = 20000

Base value for display indexes of very important tabs.

Used in *FilePropertyDialog::addTabFactory*.

**const** int **REGISTER\_FILEPROPERTYDIALOGTAB\_INDEX\_IMPORTANT** = 30000

Base value for display indexes of important tabs.

Used in *FilePropertyDialog::addTabFactory*.

**const** int **REGISTER\_FILEPROPERTYDIALOGTAB\_INDEX\_NORMAL** = 40000

Base value for display indexes of tabs with medium importance.

Used in *FilePropertyDialog::addTabFactory*.

**const** int **REGISTER\_FILEPROPERTYDIALOGTAB\_INDEX\_EXOTIC** = 50000

Base value for display indexes of exotic (i.e. less important) tabs.

Used in *FilePropertyDialog::addTabFactory*.

### 10.1.221 Class `sh::scripting::api::ApiFilePropertyDialogTab::PropertyConfig`

```
class sh::scripting::api::ApiFilePropertyDialogTab::PropertyConfig
```

#### Public Types

```
typedef QPair<QString, QString> ButtonConfig
```

#### Public Functions

```
PropertyConfig (QString title, int propertytype, QList<ButtonConfig> buttons)
```

#### Public Members

```
QString title
```

```
int propertytype
```

```
QList<ButtonConfig> buttons
```

### 10.1.222 Class `sh::scripting::api::ApiFilesystemHandler`

```
class sh::scripting::api::ApiFilesystemHandler : public sh::filesystem::FilesystemHandler
```

#### Public Functions

```
ApiFilesystemHandler (sh::filesystem::FilesystemModel *model)
```

```
void itemList (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
               sh::filesystem::FilesystemNodeType type, sh::filesystem::FilesystemNodeListEditor
               *list)
```

```
void configureItems (sh::filesystem::Operation *op, QList<std::shared_ptr<sh::filesystem::FilesystemNode>>
                     nodes)
```

```
sh::filesystem::FilesystemNodeType getType (sh::filesystem::Operation *op, std::shared_ptr<const
                                             sh::filesystem::Eurl> eurl)
```

```
qint64 getSize (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl)
```

```
QDateTime getMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl)
```

```
void setMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
               QDateTime time)
```

```
QString getMimeType (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl)
```

```
QString getLinkTarget (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                      eurl)
```

```
QList<QString> listExtendedAttributes (sh::filesystem::Operation *op, std::shared_ptr<const
                                       sh::filesystem::Eurl> eurl)
```

```
quint64 getExtendedAttributeSize (sh::filesystem::Operation *op, std::shared_ptr<const
                                  sh::filesystem::Eurl> eurl, QString attribute)
```

```
QByteArray getExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const  
                                sh::filesystem::Eurl> eurl, QString attribute)  
void setExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const  
                                sh::filesystem::Eurl> eurl, QString attribute, QByteArray value)  
void removeExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const  
                                sh::filesystem::Eurl> eurl, QString attribute)  
QMap<QString, QString> getCustomAttributes (sh::filesystem::Operation *op,  
                                                std::shared_ptr<const sh::filesystem::Eurl>  
                                                eurl)  
void setCustomAttribute (sh::filesystem::Operation *op, std::shared_ptr<const  
                                sh::filesystem::Eurl> eurl, QString attribute, QString value)  
bool canGetFileContent (sh::filesystem::Operation *op, std::shared_ptr<const  
                                sh::filesystem::Eurl> eurl)  
std::shared_ptr<QIODevice> getFileContent (sh::filesystem::Operation *op,  
                                                std::shared_ptr<const sh::filesystem::Eurl> eurl)  
bool canCreateDirectory (sh::filesystem::Operation *op, std::shared_ptr<const  
                                sh::filesystem::Eurl> eurl)  
void createDirectory (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>  
                                eurl)  
bool canCreateLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>  
                                eurl)  
void createLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,  
                                QString target)  
bool canCreateFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>  
                                eurl)  
void createFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,  
                                QIODevice *content, HandlerTransfer *handlertransfer)  
bool canDeleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>  
                                eurl)  
void deleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl)  
bool canRenameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>  
                                src)  
void renameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> src,  
                                QString destpath)  
QList<std::shared_ptr<sh::actions::AbstractActionItem>> getActions (const  
                                                                QList<std::shared_ptr<const  
                                                                sh::filesystem::Eurl>> eurls)  
void itemlist (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,  
                                sh::filesystem::FilesystemNodeType type, sh::filesystem::FilesystemNodeListEditor  
                                *list) = 0  
    Determine a list of subelements in a certain directory.  
void configureItems (sh::filesystem::Operation *op, QList<std::shared_ptr<FilesystemNode>>  
                                nodes)  
    Configure newly created nodes (e.g. setting another icon or changing the display name).  
sh::filesystem::FilesystemNodeType getType (sh::filesystem::Operation *op, std::shared_ptr<const  
                                sh::filesystem::Eurl> eurl) = 0  
    Determine if a file is regular, or a dir, or a link, ...
```

**qint64 getSize** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0  
 Determine the size of a file.

**QDateTime getMtime** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0  
 Determine the mtime of a file.

**void setMtime** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QDateTime time) = 0  
 Set the mtime of a file.

**QString getMimeType** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0  
 Determine mime type of a file.

**QString getLinkTarget** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0  
 Resolve a link.

**QList<QString> listExtendedAttributes** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)  
 Fetches the list of available extended attributes for an entry.

**qint64 getExtendedAttributeSize** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QString attribute)  
 Returns the size of value for one extended attribute for an entry.

**QByteArray getExtendedAttribute** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QString attribute)  
 Returns the value for one extended attribute for an entry.

**void setExtendedAttribute** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QString attribute, QByteArray value)  
 Sets the value for one extended attribute for an entry.

**void removeExtendedAttribute** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QString attribute)  
 Removes one extended attributes for an entry.

**QMap<QString, QString> getCustomAttributes** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)  
 Returns the custom attributes for an entry.  
  
 In contrast to extended attributes, the custom attributes are not directly stored in the filesystem this way, but handle implementation specific aspects (like permissions).

**void setCustomAttribute** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QString attribute, QString value)  
 Sets the value for one custom attribute for an entry.  
  
 See also getCustomAttributes.

**bool canGetFileContent** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0  
 Returns if it is allowed to get the content of a certain file.

**std::shared\_ptr<QIODevice> getFileContent** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0  
 Get the content of a file.

**bool canCreateDirectory** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0  
 Returns if it is allowed to create subdirectories in a certain directory.

```
void createDirectory (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl) = 0
    Create a directory.

bool canCreateLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl) = 0
    Returns if it is allowed to create a link in a certain directory.

void createLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
                 QString target) = 0
    Create a link.

bool canCreateFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl) = 0
    Returns if it is allowed to create files in a certain directory.

void createFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
                 QIODevice *content, HandlerTransfer *handlertransfer) = 0
    Creates a file with some content.

    It may use handlertransfer for some better integration, like cancel support and progress monitoring.

bool canDeleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl) = 0
    Returns if it is allowed to delete a certain file/directory/link/...

void deleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl) =
    0
    Delete a file/directory/link/...

void deleteDirectoryIfEmpty (sh::filesystem::Operation *op, std::shared_ptr<const
                             sh::filesystem::Eurl> eurl)
    Deletes a directory only if it is empty.

bool canRenameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    src) = 0
    Returns if it is allowed to rename a certain file, directory, link, ... (see renameItem).

void renameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> src,
                 QString destpath) = 0
    Renames an item to another path.

    It can be a simple filename change or also changes in the deeper path segments.

QList<std::shared_ptr<sh::actions::AbstractActionItem>> getActions (const
                                                                QList<std::shared_ptr<const
                                                                sh::filesystem::Eurl>> eurls)
                                                                = 0
    Returns a list of actions (see former example) for showing for certain files.

bool requiresResolvedLinks ()
    Returns if this handler requires to be used by the engine with resolved links.

void viewEnteredDirectory (std::shared_ptr<const sh::filesystem::Eurl>)
    Callback for preparing stuff when the view entered some directory (e.g. for enabling watchers). See also
    viewLeftDirectory.

void viewLeftDirectory (std::shared_ptr<const sh::filesystem::Eurl>)
    Callback for preparing stuff when the view left some directory. See also viewEnteredDirectory.

bool isWellKnownScheme ()
    Returns if the scheme for this handler is a well known one (which external programs typically would
    understand).
```

```
void search (sh::filesystem::Operation *op, std::shared_ptr<const
sh::filesystem::Eurl> eurl, std::function<void> std::shared_ptr<const
sh::filesystem::Eurl>, QList<std::shared_ptr<sh::search::SearchCriterion>>,
sh::filesystem::FilesystemNodeType ftype
> addfile, std::function<void>std::shared_ptr<const sh::filesystem::Eurl>> visitdir,
QList<std::shared_ptr<sh::search::SearchCriterion>> criteria)
```

Helps searching for files.

Override this method with a behavior identical to the default, if a specialized implementation can be much faster than the default one.

```
void customizeUi (std::shared_ptr<sh::filesystem::FilesystemNode> node, bool *showSearchPanel)
```

Creates a custom gui, which becomes part of the fileview.

```
sh::filesystem::FilesystemModel *model ()
```

A convenience shortcut to the *sh::filesystem::FilesystemModel* (a singleton).

## Public Members

```
std::function<int (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl) > _getType
```

```
std::function<qint64 (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl) >
_getSize
```

```
std::function<double (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl) >
_getMtime
```

```
std::function<void (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl, double) >
_setMtime
```

```
std::function<QString (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl) >
_getLinkTarget
```

```
std::function< QList< QString >sh::filesystem::Operation *op, std::shared_ptr< sh::fil
```

```
std::function<quint64 (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl, QString
attribute) > _getExtendedAttributeSize
```

```
std::function<QByteArray (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl,
QString attribute) > _getExtendedAttribute
```

```
std::function<void (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl, QString at-
tribute, QByteArray value) > _setExtendedAttribute
```

```
std::function<void (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl, QString at-
tribute) > _removeExtendedAttribute
```

```
std::function< QList< QString >sh::filesystem::Operation *op, std::shared_ptr< sh::fil
```

```
std::function<void (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl, QString at-
tribute, QString value) > _setCustomAttribute
```

```
std::function<QString (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl) >
_getMimetype
```

```
std::function<bool (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl) >
_canCreateDirectory
```

```
std::function<void (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl) >
_createDirectory
```

```
std::function<bool (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl) >
_canCreateLink
```

```
std::function<void (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl, QString tar-
get) > _createLink
```



```
std::function<bool (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl) >
    _canCreateFile
std::function<bool (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl) >
    _canDeleteItem
std::function<void (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl) >
    _deleteItem
std::function<bool (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> src) >
    _canRenameItem
std::function<void (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> src, QString dest-
    path) > _renameItem
std::function<void (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl, int type,
    sh::filesystem::FilesystemNodeListEditor *list) > _itemlist
std::function<void (sh::filesystem::Operation *op, QList<std::shared_ptr<sh::filesystem::FilesystemNode>>) >
    _configureItems
std::function<bool (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl) >
    _canGetFileContent
std::function< std::shared_ptr< QIODevice > sh::filesystem::Operation *op, std::shared_ptr<
    sh::filesystem::Eurl> eurl, QIODevice
    *content, HandlerTransfer *handlertransfer) > _createFile
std::function< QList< std::shared_ptr< sh::actions::AbstractActionItem > > QList< std::shared_ptr<
    sh::actions::AbstractActionItem > > >
```

### 10.1.223 Class sh::scripting::api::ApiFilesystemOperationProgressMonitor

```
class sh::scripting::api::ApiFilesystemOperationProgressMonitor : public sh::filesystem::FilesystemOperationProgressMonitor
```

#### Public Functions

```
ApiFilesystemOperationProgressMonitor (sh::actions::ActionExecutionInfo *actionExecution)
```

```
bool hasItemInfo ()
```

Checks if this progress monitor provides information about how many items of a certain total number are transferred so far.

```
quint64 doneItems ()
```

Checks how many items are transferred so far.

```
quint64 allItems ()
```

Checks how many items are to be transferred in total (as predicted in the current moment).

```
bool hasBytesInfo ()
```

Checks if this progress monitor provides information about how many byte of a certain total number are transferred so far.

```
quint64 doneBytes ()
```

Checks how many bytes are transferred so far.

```
quint64 allBytes ()
```

Checks how many bytes are to be transferred in total (as predicted in the current moment).

```
QString getItemInfoFrom ()
```

Returns the current source of transfer (as textual information).



QString **getItemInfoTo** ()

Returns the current destination of transfer (as textual information).

QString **estimation** ()

Returns the current performance and time estimation (as textual information).

## Public Members

std::function<void ()> **\_changed**

std::function<bool (QList<sh::filesystem::FileSystemOperationTransfers::OperationStep\*>)>  
**\_resolveConflicts**

## 10.1.224 Class sh::scripting::api::ApiGlobalObject

```
class sh::scripting::api::ApiGlobalObject : public sh::base::Singleton
```

## Public Functions

std::shared\_ptr<const sh::filesystem::Eurl> **getEurlFromString** (QString *eurl*)

void **logDebug** (QString *msg*)

void **logInfo** (QString *msg*)

void **logWarning** (QString *msg*)

void **logError** (QString *msg*)

int **filesystemnodetype\_file** ()

int **filesystemnodetype\_directory** ()

int **filesystemnodetype\_link** ()

int **filesystemnodetype\_unknown** ()

int **filesystemnodetype\_firsttype** ()

int **filesystemnodetype\_lastphysicaltype** ()

int **filesystemnodetype\_none** ()

int **filesystemnodetype\_specialtreeonlydirectory** ()

int **filesystemnodetype\_lasttype** ()

int **actiondefaultprecedencevalues\_openfile** ()

int **actiondefaultprecedencevalues\_builtinspecialopen** ()

int **messageboxbutton\_ok** ()

int **messageboxbutton\_continue** ()

int **messageboxbutton\_cancel** ()

int **messageboxbutton\_retry** ()

int **messageboxbutton\_yes** ()

int **messageboxbutton\_no** ()

int **displayindex\_core** ()

```
int displayindex_interesting()
int displayindex_exotic()
int configurationcategory_gui()
int configurationcategory_behavior()
int configurationcategory_externaltools()
int settinggroup_global()
int settinggroup_gui()
int settinggroup_filehandling()
int settinggroup_dataview()
int settinggroup_behavior()
int settinggroup_special()
int paneldetail_positionindex_veryinteresting()
int paneldetail_positionindex_interesting()
int paneldetail_positionindex_exotic()
int operationstep_conflictresolution_mergeddirectories()
int operationstep_conflictresolution_overwritedestination()
int operationstep_conflictresolution_renamedestinationbefore()
int operationstep_conflictresolution_skip()
int operationstep_conflictresolution_unresolved()
int operationstep_conflictresolution_indirect()
int operationstep_conflictresolution_usedifferentdestinationname()
int filepropertydialogtab_index_core()
int filepropertydialogtab_index_veryimportant()
int filepropertydialogtab_index_important()
int filepropertydialogtab_index_normal()
int filepropertydialogtab_index_exotic()
int filepropertydialogtab_propertytype_string()
int filepropertydialogtab_propertytype_stringmap()
int filepropertydialogtab_propertytype_icontextbanner()
int searchcriterionfactory_index_core()
int searchcriterionfactory_index_normal()
int searchcriterionfactory_index_exotic()
int thumbnailprovider_index_core()
int thumbnailprovider_index_normal()
int thumbnailprovider_index_exotic()
int thumbnailprovider_index_fallback()
```

```

void register_predicatedactionfactory (std::shared_ptr<sh::scripting::api::ApiActionFactory>
                                     f)

bool isDebugBuild ()

std::shared_ptr<sh::scripting::api::ApiFilesystemHandler> create_FilesystemHandler ()

std::shared_ptr<sh::scripting::api::ApiActionActionItem> create_ActionActionItem (QString
                                     text, bool
                                     enabled,
                                     QString
                                     icon, int
                                     defaultAc-
                                     tionPrece-
                                     dence)

std::shared_ptr<sh::scripting::api::ApiSubmenuItem> create_SubmenuItem (QString
                                     text,
                                     bool
                                     en-
                                     abled,
                                     QString
                                     icon,
                                     int
                                     default-
                                     Action-
                                     Prece-
                                     dence)

std::shared_ptr<sh::scripting::api::ApiDetailColumn> create_DetailColumn (QString      name,
                                     QString      display-
                                     Name,      int      dis-
                                     playIndex,      bool
                                     sort_doTypediff, int
                                     defaultWidth, bool
                                     isRightAligned)

std::shared_ptr<sh::scripting::api::ApiFilePropertyDialogTab> create_FilePropertyDialogTab (QString
                                     ti-
                                     tle,
                                     QList<QString>
                                     prop-
                                     er-
                                     ties)

std::shared_ptr<sh::scripting::api::ApiFilePropertyDialogTabFactory> create_FilePropertyDialogTabFactory ()

std::shared_ptr<sh::scripting::api::ApiActionFactory> create_ActionFactory (QString category,
                                     QList<std::shared_ptr<actions::Predicate>>
                                     predicates)

std::shared_ptr<sh::scripting::api::ApiPanelDetailFactorySingle> create_DetailFactorySingle (int
                                     po-
                                     si-
                                     tion,
                                     int
                                     val-
                                     ueWidth-
                                     Hint)

```

```
std::shared_ptr<sh::scripting::api::ApiPanelDetailFactoryMulti> create_DetailFactoryMulti (int
                                                    po-
                                                    si-
                                                    tion,
                                                    int
                                                    val-
                                                    ueWidth-
                                                    Hint)

std::shared_ptr<sh::scripting::api::ApiFilesystemOperationProgressMonitor> create_FilesystemOperationProgress

std::shared_ptr<sh::filesystem::FilesystemNode> _createFilesystemNode (std::shared_ptr<sh::filesystem::Eurl>
                                                                    eurl,
                                                                    sh::scripting::api::ApiFilesystemHandler
                                                                    *handler, int nodetype,
                                                                    std::shared_ptr<sh::filesystem::FilesystemNode>
                                                                    parent, bool doinsert,
                                                                    bool showInitial-
                                                                    LoadingLabel, bool
                                                                    isHidden)

std::shared_ptr<sh::scripting::api::ApiThumbnailProvider> create_ThumbnailProvider ()

std::shared_ptr<sh::filesystem::FilesystemNode> _getOrCreateFilesystemNode (std::shared_ptr<sh::filesystem::Eurl>
                                                                    eurl,
                                                                    sh::scripting::api::ApiFilesystemHandle
                                                                    *handler,
                                                                    int nodetype,
                                                                    std::shared_ptr<sh::filesystem::FilesystemNode>
                                                                    parent, bool
                                                                    doinsert, bool
                                                                    showInitial-
                                                                    LoadingLabel,
                                                                    bool isHidden)

void _register_FilesystemHandler (std::shared_ptr<sh::scripting::api::ApiFilesystemHandler>
                                handler, QString scheme)

std::shared_ptr<sh::filesystem::FilesystemNode> modelRootNode ()

QList<std::shared_ptr<sh::filesystem::FilesystemNode>> _findFilesystemNodesForEurl (std::shared_ptr<const
                                                                    sh::filesystem::Eurl>
                                                                    eurl)

QList<std::shared_ptr<sh::filesystem::FilesystemNode>> _tryGetFilesystemNodesForEurl (std::shared_ptr<const
                                                                    sh::filesystem::Eurl>
                                                                    eurl)

void refreshData (std::shared_ptr<const sh::filesystem::Eurl> eurl, bool forceFindParent, bool
                withDetails)

std::shared_ptr<filesystem::Eurl> createEurl (QString scheme, QString hostname, QString path)

std::shared_ptr<sh::filesystem::Operation> createOperation ()

sh::scripting::StringMap<QString> createArgumentException ()
```

```

sh::scripting::StringMap<QString> createIOException ()
sh::scripting::StringMap<QString> createRuntimeException ()
sh::scripting::StringMap<QString> createProgramException ()
sh::scripting::StringMap<QString> createException ()
std::shared_ptr<sh::scripting::api::ApiReadDataDevice> _createReadDataDevice ()
std::shared_ptr<ApiSetting> createSetting (QString name, QString description, int group, bool
                                         isAdvancedSetting, bool isGlobal, bool isPerFileview)
std::shared_ptr<sh::scripting::api::ApiThread> create_Thread ()
std::shared_ptr<sh::scripting::api::ApiTimer> create_Timer ()
std::shared_ptr<sh::scripting::api::ApiSearchCriterion> create_SearchCriterion (std::shared_ptr<sh::scripting::api::Api
                                                                              factory)
std::shared_ptr<sh::scripting::api::ApiSearchCriterionFactory> create_SearchCriterionFactory (QString
                                                                                             key,
                                                                                             QString
                                                                                             de-
                                                                                             scrip-
                                                                                             tion)

sh::ui::MainWindow *mainwindow ()
void registerPanelDetailFactorySingle (std::shared_ptr<sh::scripting::api::ApiPanelDetailFactorySingle>)
void registerPanelDetailFactoryMulti (std::shared_ptr<sh::scripting::api::ApiPanelDetailFactoryMulti>)
void registerSetting (std::shared_ptr<sh::settings::Setting> setting)
void openItems (QList<std::shared_ptr<const sh::filesystem::Eurl>> eurls)
void registerFilePropertyDialogTabFactory (int index, std::shared_ptr<sh::scripting::api::ApiFilePropertyDialogT
void registerThumbnailProvider (int idx, std::shared_ptr<sh::scripting::api::ApiThumbnailProvider>
                                thumbnailprovider)
void registerSearchCriterionFactory (int idx, std::shared_ptr<ApiSearchCriterionFactory>
                                    apicrit)
std::shared_ptr<sh::configuration::ConfigurationValue> register_ConfigurationValueInt (QString
                                                                                       name,
                                                                                       int
                                                                                       de-
                                                                                       flt,
                                                                                       QString
                                                                                       desc,
                                                                                       int
                                                                                       cat-
                                                                                       e-
                                                                                       gory,
                                                                                       QString
                                                                                       longdesc,
                                                                                       QString
                                                                                       change-
                                                                                       hint)

```

```
std::shared_ptr<sh::configuration::ConfigurationValue> register_ConfigurationValueFloat (QString
                                                    name,
                                                    int
                                                    de-
                                                    flt,
                                                    QString
                                                    desc,
                                                    int
                                                    cat-
                                                    e-
                                                    gory,
                                                    QString
                                                    longdesc,
                                                    QString
                                                    change-
                                                    hint)

std::shared_ptr<sh::configuration::ConfigurationValue> register_ConfigurationValueBool (QString
                                                    name,
                                                    bool
                                                    de-
                                                    flt,
                                                    QString
                                                    desc,
                                                    int
                                                    cat-
                                                    e-
                                                    gory,
                                                    QString
                                                    longdesc,
                                                    QString
                                                    change-
                                                    hint)

std::shared_ptr<sh::configuration::ConfigurationValue> register_ConfigurationValueString (QString
                                                    name,
                                                    QString
                                                    de-
                                                    flt,
                                                    QString
                                                    desc,
                                                    int
                                                    cat-
                                                    e-
                                                    gory,
                                                    QString
                                                    longdesc,
                                                    QString
                                                    change-
                                                    hint)

void registerTransferrableDetailColumn (int i, std::shared_ptr<sh::filesystem::DetailColumn>
                                         detailColumn)

std::shared_ptr<sh::filesystem::DetailColumn> findDetailColumnByName (QString name)

QString shallotDataDir ()
```

```

QString shallotBuiltinPluginDir ()
QString shallotSystemPluginDir ()
QString shallotUserPluginDir ()
QString shallotLanguage ()
sh::tools::BookmarkManager *bookmarkManager ()
std::shared_ptr<sh::actions::Predicate> create_OnDirectoriesPredicate ()
std::shared_ptr<sh::actions::Predicate> create_OnFilesPredicate ()
std::shared_ptr<sh::actions::Predicate> create_OnLinksPredicate ()
std::shared_ptr<sh::actions::Predicate> create_OnSingleEntrySelectionPredicate ()
std::shared_ptr<sh::actions::Predicate> create_DontResolveLinksPredicate ()
std::shared_ptr<sh::actions::Predicate> create_HideOnCurrentDirectoryLevelPredicate ()
std::shared_ptr<sh::actions::Predicate> create_HideOnSelectionLevelPredicate ()
std::shared_ptr<sh::actions::Predicate> create_ByRegExpPredicate (QString pattern)
std::shared_ptr<sh::actions::Predicate> create_KeyShortcutPredicate (QString shortcut, bool
                                                                    triggersOnCurrentDi-
                                                                    rectoryLevel)
std::shared_ptr<sh::actions::Predicate> create_PositionIndexPredicate (int index)
void doInitialize ()
    Executes singleton initialization.
void doShutdown ()
    Executes singleton shutdown.
void shutdown ()
    Shutdown down this singleton.
bool isAlive ()
    Returns if this singleton is alive (true until its shutdown begins).

```

## Private Functions

```
ApiGlobalObject ()
```

### 10.1.225 Class *sh::scripting::api::ApiHelperMethods*

```
class sh::scripting::api::ApiHelperMethods
```

## Public Static Functions

```
QByteArray QIODevice_read (QIODevice *self)
QByteArray QIODevice_readall (QIODevice *self)
QByteArray ApiReadDataDevice_read (sh::scripting::api::ApiReadDataDevice *self)
QByteArray ApiReadDataDevice_readall (sh::scripting::api::ApiReadDataDevice *self)
void FilesystemNodeList_resetItems (sh::filesystem::FilesystemNodeList *self, int type,
                                     QList<std::shared_ptr<sh::filesystem::FilesystemNode>>
                                     list)
void FilesystemNode_addDetail (filesystem::FilesystemNode *self,
                               std::shared_ptr<sh::filesystem::DetailColumn> col)
void FilesystemNode_setIcon (sh::filesystem::FilesystemNode *self, QString icon)
void FilesystemNode_setDisplayName (filesystem::FilesystemNode *self, QString display-
                                     name)
bool FilesystemNode_isHidden (filesystem::FilesystemNode *self)
void FilesystemNode_setHidden (filesystem::FilesystemNode *self, bool v)
void ConfigurationValue_setConfigValueInt (sh::configuration::ConfigurationValue *self,
                                             int v)
void ConfigurationValue_setConfigValueFloat (sh::configuration::ConfigurationValue
                                              *self, double v)
void ConfigurationValue_setConfigValueBool (sh::configuration::ConfigurationValue
                                             *self, bool v)
void ConfigurationValue_setConfigValueString (sh::configuration::ConfigurationValue
                                              *self, QString v)
QList<std::shared_ptr<sh::filesystem::Eurl>> FilesystemOperation_itemlist (sh::filesystem::FilesystemOperation
                                                                           *self,
                                                                           std::shared_ptr<const
                                                                           sh::filesystem::Eurl>
                                                                           eurl)
QList<std::shared_ptr<sh::filesystem::Eurl>> FilesystemOperation_itemlistByType (sh::filesystem::FilesystemOperation
                                                                           *self,
                                                                           std::shared_ptr<const
                                                                           sh::filesystem::Eurl>
                                                                           eurl,
                                                                           int
                                                                           ntype)
int FilesystemOperation_getType (sh::filesystem::FilesystemOperation *self,
                                std::shared_ptr<const sh::filesystem::Eurl> eurl)
void FilesystemOperation_createFile (sh::filesystem::FilesystemOperation *self,
                                     std::shared_ptr<const sh::filesystem::Eurl> eurl,
                                     sh::scripting::api::ApiReadDataDevice *content,
                                     std::shared_ptr<filesystem::FilesystemOperationProgressMonitor>
                                     progressmon)
QList<std::shared_ptr<sh::filesystem::FilesystemNode>> FilesystemOperation_resolveNodeLink (sh::filesystem::File
                                                                           *self,
                                                                           std::shared_ptr<sh
                                                                           node>
```



```

QList<std::shared_ptr<sh::filesystem::FilesystemNode>> FilesystemOperation_resolveNodeLink_NonRecursive

std::shared_ptr<sh::filesystem::Eurl> FilesystemOperation_resolveEurlLink (sh::filesystem::FilesystemOperation
                                                                    *self,
                                                                    std::shared_ptr<const
                                                                    sh::filesystem::Eurl>
                                                                    eurl)

std::shared_ptr<sh::filesystem::Eurl> FilesystemOperation_resolveEurlLink_NonRecursive (sh::filesystem::File
                                                                    *self,
                                                                    std::shared_ptr<const
                                                                    sh::filesystem::Eurl>
                                                                    eurl)

void PanelDetail_setRow (sh::paneldetails::PanelDetail *self, QString key, QList<QString> value)
void MainWindow_jumpToEurl (sh::ui::MainWindow *self, std::shared_ptr<const
                                                                    sh::filesystem::Eurl> eurl)

std::shared_ptr<sh::filesystem::FilesystemNode> MainWindow_getCurrentDirectoryNode (sh::ui::MainWindow
                                                                    *self)

void ApiActionActionItem_setEnabled (sh::scripting::api::ApiActionActionItem *self, bool en-
                                                                    abled)

bool ApiActionActionItem_enabled (sh::scripting::api::ApiActionActionItem *self)

void ApiActionActionItem_setVisible (sh::scripting::api::ApiActionActionItem *self, bool
                                                                    visible)

bool ApiActionActionItem_visible (sh::scripting::api::ApiActionActionItem *self)

void ApiSubmenuItem_setSubitems (sh::scripting::api::ApiSubmenuItem *self,
                                                                    QList<std::shared_ptr<actions::AbstractActionItem>>
                                                                    subitems)

void ApiSubmenuItem_setEnabled (sh::scripting::api::ApiSubmenuItem *self,
                                                                    bool enabled)

bool ApiSubmenuItem_enabled (sh::scripting::api::ApiSubmenuItem *self)

void ApiSubmenuItem_setVisible (sh::scripting::api::ApiSubmenuItem *self,
                                                                    bool visible)

bool ApiSubmenuItem_visible (sh::scripting::api::ApiSubmenuItem *self)

int OperationStep_conflictResolution (filesystem::FilesystemOperationTransfers::OperationStep
                                                                    *self)

bool FilesystemOperationProgressMonitor_hasItemInfo (sh::scripting::api::ApiFilesystemOperationProgressM
                                                                    *self)

quint64 FilesystemOperationProgressMonitor_doneItems (sh::scripting::api::ApiFilesystemOperationProgressM
                                                                    *self)

quint64 FilesystemOperationProgressMonitor_allItems (sh::scripting::api::ApiFilesystemOperationProgressM
                                                                    *self)

bool FilesystemOperationProgressMonitor_hasBytesInfo (sh::scripting::api::ApiFilesystemOperationProgress
                                                                    *self)

quint64 FilesystemOperationProgressMonitor_doneBytes (sh::scripting::api::ApiFilesystemOperationProgressM
                                                                    *self)

```

```
quint64 FilesystemOperationProgressMonitor_allBytes (sh::scripting::api::ApiFilesystemOperationProgressMonitor
                                                    *self)

QString FilesystemOperationProgressMonitor_getItemInfoFrom (sh::scripting::api::ApiFilesystemOperationProgressMonitor
                                                            *self)

QString FilesystemOperationProgressMonitor_getItemInfoTo (sh::scripting::api::ApiFilesystemOperationProgressMonitor
                                                            *self)

QString FilesystemOperationProgressMonitor_estimation (sh::scripting::api::ApiFilesystemOperationProgressMonitor
                                                         *self)

QList<std::shared_ptr<sh::filesystem::FilesystemNode>> ApiFilePropertyDialogTab_nodes (sh::scripting::api::ApiFilePropertyDialogTab
                                                                                       *self)

int ActionExecutionUserFeedback_messageBox (sh::actions::ActionExecutionUserFeedback
                                             *self, QString m, QList<QString> l, QString
                                             s, int i, int j)

QString ActionExecutionUserFeedback_simpleInputDialog (sh::actions::ActionExecutionUserFeedback
                                                         *self, QString text, QString de-
                                                         flt)

int FilePropertyDialogTab_selectedIndexForProperty (ApiFilePropertyDialogTab
                                                    *self, quintptr widget)

void FilePropertyDialogTab_refresh (ApiFilePropertyDialogTab *self)
```

### 10.1.226 Class `sh::scripting::api::ApiPanelDetailFactoryMulti`

```
class sh::scripting::api::ApiPanelDetailFactoryMulti : public sh::paneldetails::PanelDetailFactoryForFunction
```

#### Public Functions

```
ApiPanelDetailFactoryMulti (int position, int valueWidthHint)

void setvalue (std::shared_ptr<sh::paneldetails::PanelDetail> detail,
               QList<std::shared_ptr<sh::filesystem::FilesystemNode>> nodes, filesystem::Operation
               *op)

void linktriggered (QList<std::shared_ptr<sh::filesystem::FilesystemNode>> nodes, QString link-
                    target)

std::shared_ptr<PanelDetail> construct (QList<std::shared_ptr<sh::filesystem::FilesystemNode>>
                                       nodes, sh::filesystem::Operation *op)
```

#### Public Members

```
std::function<void (sh::paneldetails::PanelDetail*, QList<std::shared_ptr<sh::filesystem::FilesystemNode>>,
                   sh::filesystem::Operation *op) > _setvalue

std::function<void (QList<std::shared_ptr<sh::filesystem::FilesystemNode>>,
                   QString) > _linktriggered
```

### 10.1.227 Class `sh::scripting::api::ApiPanelDetailFactorySingle`

```
class sh::scripting::api::ApiPanelDetailFactorySingle : public sh::paneldetails::PanelDetailFactoryForFun
```

#### Public Functions

**ApiPanelDetailFactorySingle** (int *position*, int *valueWidthHint*)

void **setvalue** (std::shared\_ptr<sh::paneldetails::PanelDetail>, std::shared\_ptr<sh::filesystem::FilesystemNode>  
*node*, sh::filesystem::Operation \**op*)

void **linktriggered** (std::shared\_ptr<sh::filesystem::FilesystemNode> *node*, QString *linktarget*)

std::shared\_ptr<PanelDetail> **construct** (QList<std::shared\_ptr<sh::filesystem::FilesystemNode>>  
*nodes*, sh::filesystem::Operation \**op*)

#### Public Members

std::function<void (sh::paneldetails::PanelDetail\*, std::shared\_ptr<sh::filesystem::FilesystemNode>,  
sh::filesystem::Operation \*op) > **\_setvalue**

std::function<void (std::shared\_ptr<sh::filesystem::FilesystemNode>, QString) > **\_linktriggered**

### 10.1.228 Class `sh::scripting::api::ApiReadDataDevice`

```
class sh::scripting::api::ApiReadDataDevice : public sh::tools::ReadDataDevice
```

#### Public Functions

**ApiReadDataDevice** ()

QByteArray **getdata** ()

This method returns the next available chunk of data. It may return empty arrays whenever temporarily no data is available. Returning a null array (with `QByteArray::isNull() == true`) marks the end of the stream. .

void **\_ended** ()

#### Public Members

std::function<QByteArray () > **\_getdata**

#### Private Members

bool **isended**

### 10.1.229 Class `sh::scripting::api::ApiSearchCriterion`

```
class sh::scripting::api::ApiSearchCriterion : public sh::search::criteria::AbstractActionDrivenSearchCriterion
```

#### Public Functions

**ApiSearchCriterion** (std::shared\_ptr<sh::search::SearchCriterionFactory> factory)

bool **match2** (sh::filesystem::Operation \*op, std::shared\_ptr<const sh::filesystem::Eurl> eurl)

QList<QString> **getsearchspec** ()

bool **match** (sh::filesystem::Operation \*op, std::shared\_ptr<const sh::filesystem::Eurl> eurl)  
Checks if a given file (by eurl) matches this criterion. .

QString **valuedescription** ()  
Returns a human readable text describing the current criterion shortly and precisely. .

QString **serialize** ()  
Serializes this criterion to a string. See also *SearchCriterion::deserialize*.

std::shared\_ptr<sh::search::SearchCriterionFactory> **factory** ()  
Returns the factory this criterion has created (provides some more metadata).

#### Public Members

std::function<void (sh::actions::ActionExecutionInfo\*)> **\_configure**

std::function<bool (sh::filesystem::Operation \*op, std::shared\_ptr<sh::filesystem::Eurl> eurl)> **\_match**

QStringList **searchspec**

#### Public Static Functions

void **\_createEditor** (std::shared\_ptr<sh::search::SearchCriterion> *cfg*,  
sh::ui::SearchPanelConfiguration \*panelcfg)

void **\_editorUpdateConfiguration** (sh::ui::SearchPanelConfiguration \*panelcfg,  
std::shared\_ptr<sh::search::SearchCriterion> c)

std::shared\_ptr<sh::search::SearchCriterion> **deserialize** (QString s)  
Deserializes this criterion from a string. See also *SearchCriterion::serialize*.

### 10.1.230 Class `sh::scripting::api::ApiSearchCriterionFactory`

```
class sh::scripting::api::ApiSearchCriterionFactory : public sh::search::SearchCriterionFactory
```

## Public Functions

**ApiSearchCriterionFactory** (QString *key*, QString *description*)

QString **key** ()

Returns the string key for the associated search criterion class. .

QString **description** ()

Returns the description string for the associated search criterion class. .

void **editor** (std::shared\_ptr<sh::search::SearchCriterion> *c*, sh::ui::SearchPanelConfiguration \**panelcfg*)

void **editorupdateconfig** (sh::ui::SearchPanelConfiguration \**panelcfg*,  
std::shared\_ptr<sh::search::SearchCriterion> *c*)

std::shared\_ptr<sh::search::SearchCriterion> **construct** ()

Constructs an instance of the associated search criterion class. .

bool **isVisibleFor** (sh::filesystem::Operation \**op*, std::shared\_ptr<sh::filesystem::FilesystemNode>  
*node*)

Checks if the associated search criterion class should be offered to the user for a given node. .

void **editor** (std::shared\_ptr<sh::search::SearchCriterion> *c*, sh::ui::SearchPanelConfiguration \**panelcfg*) = 0

Builds the search config ui in a search panel.

Takes the data from *c* and populates *panelcfg* with it.

void **editorupdateconfig** (sh::ui::SearchPanelConfiguration \**panelcfg*,  
std::shared\_ptr<sh::search::SearchCriterion> *c*) = 0

Updates own data with the content from the search config ui in a search panel.

Takes the data from *panelcfg* and updates *c* with it.

## Public Members

std::function<QString (QList<QString>)> **\_\_getsearchspecdesc**

std::function< std::shared\_ptr< sh::search::SearchCriterion >> **\_\_construct**

std::function<bool (sh::filesystem::Operation\*, std::shared\_ptr<sh::filesystem::FilesystemNode>)>  
**\_\_isVisibleFor**

### 10.1.231 Class sh::scripting::api::ApiSetting

**class** sh::scripting::api::ApiSetting : public sh::settings::Setting

## Public Functions

**ApiSetting** (QString *name*, QString *description*, sh::settings::SettingGroup *group*, bool *isAdvancedSetting*, bool *isGlobal*, bool *isPerFileview*)

QString **name** ()

Gets the internal name.

QString **description** ()

Gets the description text.

*sh::settings::SettingGroup* **group** ()  
Gets the group;.

bool **isAdvancedSetting** ()  
Is this an advanced setting?

void **setValue** (QString *value*)  
Called from Shallot core when the value was set (for a not-per-fileview setting).

void **setValue** (*sh::ui::FileView* \**filelist*, QString *value*)  
Called from Shallot core when the value was set (for a per-fileview setting).

QString **getValue** (*sh::ui::FileView* \**filelist*)  
Get the currently set value.

bool **isGlobal** ()  
Does this setting apply globally or just for a certain subtree of nodes?

bool **isPerFileview** ()  
Does this setting apply for each fileview individually or for the complete main window?

QString **valueDescription** (QString *value*)  
Gets a human readable description text for a value.

## Public Members

std::function<void (QString) > **\_setValue1**

std::function<void (int, QString) > **\_setValue2**

std::function<QString (int) > **\_getValue**

std::function<QString (QString) > **\_valueDescription**

QString **\_name**

QString **\_description**

*sh::settings::SettingGroup* **\_group**

bool **\_isAdvancedSetting**

bool **\_isGlobal**

bool **\_isPerFileview**

## Public Static Functions

QString **getGroupDescription** (int *g*)  
Low-level function which gets the description text of a group.

### 10.1.232 Class `sh::scripting::api::ApiSubmenuItem`

```
class sh::scripting::api::ApiSubmenuItem : public sh::actions::SubmenuItem
```

#### Public Functions

**ApiSubmenuItem** (QString *text*, bool *enabled*, QString *icon*, int *defaultActionPrecedence*)

void **initialize** () **override**

Initialize the action. This should make the time-consuming parts, e.g. for determining a label or enabled state.

**const** QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> **subitems** ()

Returns the list of subitems from this submenu.

void **setSubitems** (**const** QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> *subitems*)

Sets the subitems.

QString **text** ()

Returns the displayed text for this action.

QString **icon** ()

Returns the icon for this action.

bool **enabled** ()

Checks if this action is enabled.

bool **isChecked** ()

Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<AbstractActionItem> **parentAction** ()

Returns the parent action, if it is added to a container.

```
void _setParentAction (std::shared_ptr<AbstractActionItem> parent)  
    Sets the parent action. .  
void initializeAsync (std::function<void>  
    > oninitialized = 0) Asynchronously initializes the action.  
void initializeSync ()  
    Synchronously initializes the action.  
bool isInitialized ()  
    Checks if this action is initialized.  
bool isInitializing ()  
    Checks if this action is initializing.
```

### Public Members

```
std::function<void ()> _initialize
```

### Signals

```
void subitemsChanged ()  
    Emits when the list of subitems changed.  
void changed ()  
    Emits when some data changed.
```

## 10.1.233 Class `sh::scripting::api::ApiThread`

```
class sh::scripting::api::ApiThread : public std::enable_shared_from_this<ApiThread>
```

### Public Functions

```
ApiThread ()  
void start ()
```

### Public Members

```
std::function<void ()> _run
```

## 10.1.234 Class `sh::scripting::api::ApiThumbnailProvider`

```
class sh::scripting::api::ApiThumbnailProvider : public sh::tools::ThumbnailProvider
```



## Public Functions

**ApiThumbnailProvider** ()

void **getThumbnail** (*sh::filesystem::Operation* \*operation, std::shared\_ptr<*sh::filesystem::FilesystemNode*> node, QString contentType, int width, int height, QIcon \*icon)

## Public Members

std::function<QByteArray (*sh::filesystem::Operation*\*, std::shared\_ptr<*sh::filesystem::FilesystemNode*>, QString, int, int) > **\_getThumbnail**

### 10.1.235 Class sh::scripting::api::ApiTimer

**class** *sh::scripting::api::ApiTimer* : **public** std::enable\_shared\_from\_this<*ApiTimer*>

## Public Functions

**ApiTimer** ()

void **start** (int interval)

void **stop** ()

## Public Members

std::function<void () > **\_run**

## Private Members

QMutex **\_mutex**

QTimer **\_timer**

bool **\_isexecuting** = false

QList<std::shared\_ptr<*ApiTimer*>> **\_runningTimers**

### 10.1.236 Class sh::scripting::pythoninterop::Converters

**class** *sh::scripting::pythoninterop::Converters*

## Public Static Functions

QString **getStringFromPyObject** (PyObject \*o)

QVariant **getVariantFromPyObject** (PyObject \*o)

int **getIntFromPyObject** (PyObject \*o)

double **getDoubleFromPyObject** (PyObject \*o)

qint64 **getInt64FromPyObject** (PyObject \*o)

```
quint64 getUint64FromPyObject (PyObject *o)
bool getBoolFromPyObject (PyObject *o)
QByteArray getByteArrayFromPyObject (PyObject *o)
void *getPointerFromPyObject (PyObject *o)
std::shared_ptr<void> getSharedPointerFromPyObject (PyObject *o)
PyObject *getPyObjectFromString (QString v)
PyObject *getPyObjectFromVariant (QVariant v)
PyObject *getPyObjectFromInt (int v)
PyObject *getPyObjectFromDouble (double v)
PyObject *getPyObjectFromInt64 (qint64 v)
PyObject *getPyObjectFromUInt64 (quint64 v)
PyObject *getPyObjectFromBool (bool v)
PyObject *getPyObjectFromByteArray (QByteArray v)
```

### 10.1.237 Class `sh::scripting::pythoninterop::NativeToPyObject`

```
template<class V>
class NativeToPyObject
    Helps to return a Python object from a native c++ value (with reference ownership)
```

### 10.1.238 Class `sh::scripting::pythoninterop::NativeToPyObject< QByteArray >`

```
template<>
class sh::scripting::pythoninterop::NativeToPyObject<QByteArray>
```

#### Public Static Functions

```
PyObject *toPyObject (QByteArray v)
```

### 10.1.239 Class `sh::scripting::pythoninterop::NativeToPyObject< QList< V > >`

```
template<typename V>
class sh::scripting::pythoninterop::NativeToPyObject<QList<V>>
```

#### Public Static Functions

```
PyObject *toPyObject (QList<V> v)
```

### 10.1.240 Class `sh::scripting::pythoninterop::NativeToPyObject< QMap< QString, V > >`

```
template<typename V>
class sh::scripting::pythoninterop::NativeToPyObject<QMap<QString, V>>
```

#### Public Static Functions

```
PyObject *toPyObject (QMap<QString, V> v)
```

### 10.1.241 Class `sh::scripting::pythoninterop::NativeToPyObject< QString >`

```
template<>
class sh::scripting::pythoninterop::NativeToPyObject<QString>
```

#### Public Static Functions

```
PyObject *toPyObject (QString v)
```

### 10.1.242 Class `sh::scripting::pythoninterop::NativeToPyObject< QVariant >`

```
template<>
class sh::scripting::pythoninterop::NativeToPyObject<QVariant>
```

#### Public Static Functions

```
PyObject *toPyObject (QVariant v)
```

### 10.1.243 Class `sh::scripting::pythoninterop::NativeToPyObject< V * >`

```
template<class V>
class sh::scripting::pythoninterop::NativeToPyObject<V*>
```

#### Public Static Functions

```
PyObject *toPyObject (V *v)
```

### 10.1.244 Class `sh::scripting::pythoninterop::NativeToPyObject< bool >`

```
template<>
class sh::scripting::pythoninterop::NativeToPyObject<bool>
```

**Public Static Functions**

PyObject \***toPyObject** (bool v)

**10.1.245 Class sh::scripting::pythoninterop::NativeToPyObject< double >**

template<>

**class** *sh::scripting::pythoninterop::NativeToPyObject*<double>

**Public Static Functions**

PyObject \***toPyObject** (double v)

**10.1.246 Class sh::scripting::pythoninterop::NativeToPyObject< int >**

template<>

**class** *sh::scripting::pythoninterop::NativeToPyObject*<int>

**Public Static Functions**

PyObject \***toPyObject** (int v)

**10.1.247 Class sh::scripting::pythoninterop::NativeToPyObject< qint64 >**

template<>

**class** *sh::scripting::pythoninterop::NativeToPyObject*<qint64>

**Public Static Functions**

PyObject \***toPyObject** (qint64 v)

**10.1.248 Class sh::scripting::pythoninterop::NativeToPyObject< quint64 >**

template<>

**class** *sh::scripting::pythoninterop::NativeToPyObject*<quint64>

**Public Static Functions**

PyObject \***toPyObject** (quint64 v)

### 10.1.249 Class `sh::scripting::pythoninterop::NativeToPyObject< std::shared_ptr< V > >`

```
template<class V>
class sh::scripting::pythoninterop::NativeToPyObject<std::shared_ptr<V>>
```

#### Public Static Functions

```
PyObject *toPyObject (std::shared_ptr<V> v)
```

### 10.1.250 Class `sh::scripting::pythoninterop::NativeToPyObject< std::shared_ptr< const V > >`

```
template<class V>
class sh::scripting::pythoninterop::NativeToPyObject<std::shared_ptr<const V>>
```

#### Public Static Functions

```
PyObject *toPyObject (std::shared_ptr<const V> v)
```

### 10.1.251 Class `sh::scripting::pythoninterop::PyObjectFunctionToNativeFunction`

```
template<class R, class ...Args>
class sh::scripting::pythoninterop::PyObjectFunctionToNativeFunction
```

#### Public Static Functions

```
std::function<R (Args...)> toNativeFunction
    PyObject *self, PyObject *fct
```

### 10.1.252 Class `sh::scripting::pythoninterop::PyObjectFunctionToNativeFunction< void, Args... >`

```
template<class ...Args>
class sh::scripting::pythoninterop::PyObjectFunctionToNativeFunction<void, Args...>
```

#### Public Static Functions

```
std::function<void (Args...)> toNativeFunction
    PyObject *self, PyObject *fct
```

### 10.1.253 Class `sh::scripting::pythoninterop::PyObjectToNative`

```
template<class V>
class PyObjectToNative
    Helps to return a c++ value from a Python object (borrows the reference)
```

### 10.1.254 Class `sh::scripting::pythoninterop::PyObjectToNative< QByteArray >`

```
template<>
class sh::scripting::pythoninterop::PyObjectToNative<QByteArray>
```

#### Public Static Functions

`QByteArray toNative (PyObject*, PyObject *o)`

### 10.1.255 Class `sh::scripting::pythoninterop::PyObjectToNative< QList< V > >`

```
template<class V>
class sh::scripting::pythoninterop::PyObjectToNative<QList<V>>
```

#### Public Static Functions

`QList<V> toNative (PyObject *a, PyObject *o)`

### 10.1.256 Class `sh::scripting::pythoninterop::PyObjectToNative< QString >`

```
template<>
class sh::scripting::pythoninterop::PyObjectToNative<QString>
```

#### Public Static Functions

`QString toNative (PyObject*, PyObject *o)`

### 10.1.257 Class `sh::scripting::pythoninterop::PyObjectToNative< QVariant >`

```
template<>
class sh::scripting::pythoninterop::PyObjectToNative<QVariant>
```

#### Public Static Functions

`QVariant toNative (PyObject*, PyObject *o)`

**10.1.258 Class `sh::scripting::pythoninterop::PyObjectToNative< StringMap< V > >`**

```
template<class V>
class sh::scripting::pythoninterop::PyObjectToNative<StringMap<V>>
```

**Public Static Functions**

```
StringMap<V> toNative (PyObject *a, PyObject *o)
```

**10.1.259 Class `sh::scripting::pythoninterop::PyObjectToNative< V * >`**

```
template<class V>
class sh::scripting::pythoninterop::PyObjectToNative<V*>
```

**Public Static Functions**

```
V *toNative (PyObject*, PyObject *o)
```

**10.1.260 Class `sh::scripting::pythoninterop::PyObjectToNative< bool >`**

```
template<>
class sh::scripting::pythoninterop::PyObjectToNative<bool>
```

**Public Static Functions**

```
bool toNative (PyObject*, PyObject *o)
```

**10.1.261 Class `sh::scripting::pythoninterop::PyObjectToNative< double >`**

```
template<>
class sh::scripting::pythoninterop::PyObjectToNative<double>
```

**Public Static Functions**

```
double toNative (PyObject*, PyObject *o)
```

**10.1.262 Class `sh::scripting::pythoninterop::PyObjectToNative< int >`**

```
template<>
class sh::scripting::pythoninterop::PyObjectToNative<int>
```

**Public Static Functions**

int **toNative** (PyObject\*, PyObject \*o)

**10.1.263 Class sh::scripting::pythoninterop::PyObjectToNative< qint64 >**

template<>

**class** *sh::scripting::pythoninterop::PyObjectToNative*<qint64>

**Public Static Functions**

qint64 **toNative** (PyObject\*, PyObject \*o)

**10.1.264 Class sh::scripting::pythoninterop::PyObjectToNative< quint64 >**

template<>

**class** *sh::scripting::pythoninterop::PyObjectToNative*<quint64>

**Public Static Functions**

quint64 **toNative** (PyObject\*, PyObject \*o)

**10.1.265 Class sh::scripting::pythoninterop::PyObjectToNative< std::function< V(Args...)> >**

template<class V, class ...Args>

**class** *sh::scripting::pythoninterop::PyObjectToNative*<std::function<V (Args...) >>

**Public Static Functions**

std::function<V (Args...) > **toNative**  
PyObject \*self, PyObject \*io

**10.1.266 Class sh::scripting::pythoninterop::PyObjectToNative< std::shared\_ptr< V > >**

template<class V>

**class** *sh::scripting::pythoninterop::PyObjectToNative*<std::shared\_ptr<V>>



### Public Static Functions

std::shared\_ptr<V> **toNative** (PyObject\*, PyObject \*o)

#### 10.1.267 Class sh::scripting::pythoninterop::\_PyObjectFunctionToNativeFunction\_SetTupleValue

```
template<int pos, class ...Args>
class _PyObjectFunctionToNativeFunction_SetTupleValue
```

#### 10.1.268 Class sh::scripting::pythoninterop::\_PyObjectFunctionToNativeFunction\_SetTupleValue<pos, Arg, Args...>

```
template<int pos, class Arg, class ...Args>
class sh::scripting::pythoninterop::_PyObjectFunctionToNativeFunction_SetTupleValue<pos, Arg,
```

### Public Static Functions

void **setTupleValue** (PyObject \*tup, Arg arg, Args... args)

#### 10.1.269 Class sh::scripting::pythoninterop::\_PyObjectFunctionToNativeFunction\_SetTupleValue<pos>

```
template<int pos>
class sh::scripting::pythoninterop::_PyObjectFunctionToNativeFunction_SetTupleValue<pos>
```

### Public Static Functions

void **setTupleValue** (PyObject\*)

#### 10.1.270 Class sh::scripting::pythoninterop::\_PyObjectFunctionToNativeFunction\_TupleValueSetter

```
template<int pos, class ...Args>
class sh::scripting::pythoninterop::_PyObjectFunctionToNativeFunction_TupleValueSetter
    This class is a workaround for gcc<4.9 bug 55914. It was not able to call ...::setTupleValue directly from within
    the invokePython lambda
```

### Public Functions

**\_PyObjectFunctionToNativeFunction\_TupleValueSetter** (Args... args)

void **set** (PyObject \*tuple)

### Private Members

std::function<void (PyObject\*)> **\_set**

### Private Static Functions

void **\_\_set** (*Args...* *args*, PyObject *\*tuple*)

## 10.1.271 Class `sh::search::Search`

**class** `sh::search::Search` : **public** QObject, **public** std::enable\_shared\_from\_this<Search>

A search object represents one search request the user made.

### Public Functions

**~Search** ()

std::shared\_ptr<const *sh::filesystem::Eurl*> **eurl** ()  
Returns the eurl containing the search configuration.

std::shared\_ptr<*sh::filesystem::FilesystemNode*> **searchnode** ()  
Returns the root node of the search (the one named like ‘*Search ...*’).

**Search** (std::shared\_ptr<const *filesystem::Eurl*> *eurl*, std::shared\_ptr<*sh::filesystem::FilesystemNode*>  
*searchnode*)  
See *sh::search::SearchManager::requestSearch*.

void **search** ()  
Start the searching.

bool **isSearching** ()  
Returns if the search is currently in progress.

### Signals

void **isSearchingChanged** ()  
Triggers when *Search::isSearching* changes.

### Private Functions

void **search** (std::shared\_ptr<const *filesystem::Eurl*> *eurl*, QString *relpath*)

void **insertFileItem** (std::shared\_ptr<const *sh::filesystem::Eurl*> *item*, QString *relpath*,  
*sh::filesystem::FilesystemNodeType* *ftype*)

std::shared\_ptr<*sh::filesystem::FilesystemNode*> **getDirItem** (std::shared\_ptr<const  
*sh::filesystem::Eurl*> *item*)

## Private Members

```
std::shared_ptr<const sh::filesystem::Eurl> _eurl
std::shared_ptr<sh::filesystem::FilesystemNode> _searchnode
sh::search::SearchConfiguration *_config
QHash<std::shared_ptr<const sh::filesystem::Eurl>, std::shared_ptr<sh::filesystem::FilesystemNode>> _dirs
QMutex _isSearchingMutex
bool _isSearching = false
bool _isSearchAgain = false
```

### 10.1.272 Class sh::search::SearchConfiguration

**class** *sh::search::SearchConfiguration*

A search configuration.

*Search* configurations contain a list of search criteria and some auxiliary fields. Each search configuration describes what and how a user wants to search (just not where).

## Public Functions

**SearchConfiguration()**

QString **serialize()**

Serialize this search configuration to a string. See also *SearchConfiguration::deserialize*.

## Public Members

QList<std::shared\_ptr<*sh::search::SearchCriterion*>> **criteria**

bool **showAsTree** = false

## Public Static Functions

*SearchConfiguration* \***deserialize**(QString s)

Deserialize a search configuration from a string. See also *SearchConfiguration::serialize*.

### 10.1.273 Class sh::search::SearchCriterion

**class** *sh::search::SearchCriterion*

Abstract base class for a search criterion.

Each search criterion implements how a user can filter his files in a search.

Register an implementation, e.g. with ‘REGISTER\_CRITERION\_FACTORY’ of “search/searchcriterionfactoryfromfunction.h”.

Subclassed by *sh::search::criteria::AbstractActionDrivenSearchCriterion*, *sh::search::criteria::AbstractStringSearchCriterion*, *sh::search::criteria::ExtendedAttributeSearchCriterion*, *sh::search::criteria::MtimeSearchCriterion*

## Public Functions

**SearchCriterion** (std::shared\_ptr<sh::search::SearchCriterionFactory> factory)

**~SearchCriterion** ()

QString **serialize** ()

Serializes this criterion to a string. See also *SearchCriterion::deserialize*.

bool **match** (sh::filesystem::Operation \*op, std::shared\_ptr<const sh::filesystem::Eurl> eurl) = 0

Checks if a given file (by eurl) matches this criterion. .

QString **valuedescription** () = 0

Returns a human readable text describing the current criterion shortly and precisely. .

std::shared\_ptr<sh::search::SearchCriterionFactory> **factory** ()

Returns the factory this criterion has created (provides some more metadata).

## Public Static Functions

std::shared\_ptr<sh::search::SearchCriterion> **deserialize** (QString s)

Deserializes this criterion from a string. See also *SearchCriterion::serialize*.

## 10.1.274 Class sh::search::SearchCriterionFactory

**class** sh::search::SearchCriterionFactory : public std::enable\_shared\_from\_this<SearchCriterionFactory>

Abstract base class for a search criterion factory, which constructs some *sh::search::SearchCriterion* instances.

It also provides some control methods and metadata (which is possible since there is a separate factory for each criterion class).

Note: Though possible in some exotic situations, one should not override this class. Override and register *sh::search::SearchCriterion* instead!

Subclassed by *sh::scripting::api::ApiSearchCriterionFactory*, *sh::search::SearchCriterionFactoryFromFunction*

## Public Functions

**SearchCriterionFactory** ()

QString **key** () = 0

Returns the string key for the associated search criterion class. .

QString **description** () = 0

Returns the description string for the associated search criterion class. .

void **editor** (std::shared\_ptr<sh::search::SearchCriterion> c, sh::ui::SearchPanelConfiguration \*panelcfg) = 0

Builds the search config ui in a search panel.

Takes the data from c and populates panelcfg with it.

void **editorupdateconfig** (sh::ui::SearchPanelConfiguration \*panelcfg, std::shared\_ptr<sh::search::SearchCriterion> c) = 0

Updates own data with the content from the search config ui in a search panel.

Takes the data from panelcfg and updates c with it.

std::shared\_ptr<sh::search::SearchCriterion> **construct** () = 0

Constructs an instance of the associated search criterion class. .

```
bool isVisibleFor(sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::FilesystemNode>
                 node)
    Checks if the associated search criterion class should be offered to the user for a given node. .

~SearchCriterionFactory()
```

### 10.1.275 Class *sh::search::SearchCriterionFactoryFromFunction*

**class** *sh::search::SearchCriterionFactoryFromFunction* : **public** *sh::search::SearchCriterionFactory*  
 A search criterion factory implementation which uses some additional static methods in the *sh::search::SearchCriterion* subclasses for control.

This is the default search criterion factory. It is typically used indirectly by the ‘REGISTER\_CRITERION\_FACTORY’ macro from here.

#### Public Functions

```
SearchCriterionFactoryFromFunction (QString key, std::function<QString>
    > descriptionfct, std::function<void std::shared_ptr<sh::search::SearchCriterion>,
    sh::ui::SearchPanelConfiguration*> editorfct, std::function<void sh::ui::SearchPanelConfiguration*,
    std::shared_ptr<sh::search::SearchCriterion>> editorupdateconfgct, std::function<std::shared_ptr<sh::search::SearchCrite
    constructfct

QString key ()
    Returns the string key for the associated search criterion class. .

QString description ()
    Returns the description string for the associated search criterion class. .

void editor (std::shared_ptr<sh::search::SearchCriterion> c, sh::ui::SearchPanelConfiguration *panel-
    cfg)
    Builds the search config ui in a search panel.

    Takes the data from c and populates panelcfg with it.

void editorupdateconfig (sh::ui::SearchPanelConfiguration *panelcfg,
    std::shared_ptr<sh::search::SearchCriterion> c)
    Updates own data with the content from the search config ui in a search panel.

    Takes the data from panelcfg and updates c with it.

std::shared_ptr<SearchCriterion> construct ()
    Constructs an instance of the associated search criterion class. .

bool isVisibleFor (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::FilesystemNode>
    node)
    Checks if the associated search criterion class should be offered to the user for a given node. .
```

## Private Members

QString **\_key**

std::function<QString ()> **\_descriptionfct**

std::function<void (std::shared\_ptr<sh::search::SearchCriterion>, sh::ui::SearchPanelConfiguration\*)>  
**\_editorfct**

std::function<void (sh::ui::SearchPanelConfiguration\*, std::shared\_ptr<sh::search::SearchCriterion>)>  
**\_editorupdateconfigfct**

std::function< std::shared\_ptr< sh::search::SearchCriterion >std::shared\_ptr< sh::search::SearchCriterion >>

## 10.1.276 Class sh::search::SearchFilesystemHandler

**class** *sh::search::SearchFilesystemHandler* : **public** *sh::filesystem::FilesystemHandler*, **public** *sh::base::Singleton*

A filesystem handler for presenting filesystem search results.

## Public Functions

void **itemlist** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl,  
*sh::filesystem::FilesystemNodeType* type, *sh::filesystem::FilesystemNodeListEditor*  
\*list) **override**

*sh::filesystem::FilesystemNodeType* **getType** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl*> eurl) **override**

qint64 **getSize** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)  
**override**

QDateTime **getMtime** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*>  
eurl) **override**

void **setMtime** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl,  
QDateTime time) **override**

QString **getMimeType** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*>  
eurl) **override**

QString **getLinkTarget** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*>  
eurl) **override**

bool **canGetFileContent** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl*> eurl) **override**

std::shared\_ptr<QIODevice> **getFileContent** (*sh::filesystem::Operation* \*op,  
std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)  
**override**

bool **canCreateDirectory** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl*> eurl) **override**

void **createDirectory** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*>  
eurl) **override**

bool **canCreateLink** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*>  
eurl) **override**

void **createLink** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl,  
QString target) **override**

bool **canCreateFile** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*>  
eurl) **override**

```

void createFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
                QIODevice *content, HandlerTransfer *handlertransfer) override

bool canDeleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl) override

void deleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl)
override

bool canRenameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    src) override

void renameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> src,
                 QString destpath) override

QList<std::shared_ptr<sh::actions::AbstractActionItem>> getActions (const
                                                                QList<std::shared_ptr<const
                                                                sh::filesystem::Eurl>> eurls)
                                                                override

void customizeUi (std::shared_ptr<sh::filesystem::FilesystemNode> node, bool *showSearchPanel)
override

void configureItems (sh::filesystem::Operation *op, QList<std::shared_ptr<sh::filesystem::FilesystemNode>>
                     nodes) override
    Configure newly created nodes (e.g. setting another icon or changing the display name).

void itemlist (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
               sh::filesystem::FilesystemNodeType type, sh::filesystem::FilesystemNodeListEditor
               *list) = 0
    Determine a list of subelements in a certain directory.

sh::filesystem::FilesystemNodeType getType (sh::filesystem::Operation *op, std::shared_ptr<const
                                             sh::filesystem::Eurl> eurl) = 0
    Determine if a file is regular, or a dir, or a link, ...

qint64 getSize (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl) = 0
    Determine the size of a file.

QDateTime getMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl) = 0
    Determine the mtime of a file.

void setMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
               QDateTime time) = 0
    Set the mtime of a file.

QString getMimetype (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                     eurl) = 0
    Determine mime type of a file.

QString getLinkTarget (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                       eurl) = 0
    Resolve a link.

QList<QString> listExtendedAttributes (sh::filesystem::Operation *op, std::shared_ptr<const
                                       sh::filesystem::Eurl> eurl)
    Fetches the list of available extended attributes for an entry.

quint64 getExtendedAttributeSize (sh::filesystem::Operation *op, std::shared_ptr<const
                                   sh::filesystem::Eurl> eurl, QString attribute)
    Returns the size of value for one extended attribute for an entry.

QByteArray getExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
                                   sh::filesystem::Eurl> eurl, QString attribute)
    Returns the value for one extended attribute for an entry.

```

void **setExtendedAttribute** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QString attribute, QByteArray value)  
Sets the value for one extended attribute for an entry.

void **removeExtendedAttribute** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QString attribute)  
Removes one extended attributes for an entry.

QMap<QString, QString> **getCustomAttributes** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)  
Returns the custom attributes for an entry.

In contrast to extended attributes, the custom attributes are not directly stored in the filesystem this way, but handle implementation specific aspects (like permissions).

void **setCustomAttribute** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QString attribute, QString value)  
Sets the value for one custom attribute for an entry.  
See also getCustomAttributes.

bool **canGetFileContent** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0  
Returns if it is allowed to get the content of a certain file.

std::shared\_ptr<QIODevice> **getFileContent** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0  
Get the content of a file.

bool **canCreateDirectory** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0  
Returns if it is allowed to create subdirectories in a certain directory.

void **createDirectory** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0  
Create a directory.

bool **canCreateLink** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0  
Returns if it is allowed to create a link in a certain directory.

void **createLink** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QString target) = 0  
Create a link.

bool **canCreateFile** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0  
Returns if it is allowed to create files in a certain directory.

void **createFile** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QIODevice \*content, HandlerTransfer \*handlertransfer) = 0  
Creates a file with some content.

It may use handlertransfer for some better integration, like cancel support and progress monitoring.

bool **canDeleteItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0  
Returns if it is allowed to delete a certain file/directory/link/...

void **deleteItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0  
Delete a file/directory/link/...



void **deleteDirectoryIfEmpty** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)  
 Deletes a directory only if it is empty.

bool **canRenameItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> src) = 0  
 Returns if it is allowed to rename a certain file, directory, link, ... (see renameItem).

void **renameItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> src, QString destpath) = 0  
 Renames an item to another path.  
 It can be a simple filename change or also changes in the deeper path segments.

QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> **getActions** (const QList<std::shared\_ptr<const *sh::filesystem::Eurl*>> eurls) = 0  
 Returns a list of actions (see former example) for showing for certain files.

bool **requiresResolvedLinks** ()  
 Returns if this handler requires to be used by the engine with resolved links.

void **viewEnteredDirectory** (std::shared\_ptr<const *sh::filesystem::Eurl*>)  
 Callback for preparing stuff when the view entered some directory (e.g. for enabling watchers). See also viewLeftDirectory.

void **viewLeftDirectory** (std::shared\_ptr<const *sh::filesystem::Eurl*>)  
 Callback for preparing stuff when the view left some directory. See also viewEnteredDirectory.

bool **isWellKnownScheme** ()  
 Returns if the scheme for this handler is a well known one (which external programs typically would understand).

void **search** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, std::function<void> std::shared\_ptr<const *sh::filesystem::Eurl*>, QList<std::shared\_ptr<*sh::search::SearchCriterion*>>, *sh::filesystem::FilesystemNodeType* ftype  
 > addfile, std::function<void>std::shared\_ptr<const *sh::filesystem::Eurl*>> visitdir, QList<std::shared\_ptr<*sh::search::SearchCriterion*>> criteria)  
 Helps searching for files.  
 Override this method with a behavior identical to the default, if a specialized implementation can be much faster than the default one.

void **customizeUi** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> node, bool \*showSearchPanel)  
 Creates a custom gui, which becomes part of the fileview.

*sh::filesystem::FilesystemModel* \***model** ()  
 A convenience shortcut to the *sh::filesystem::FilesystemModel* (a singleton).

void **doShutdown** ()  
 Executes singleton shutdown.

void **shutdown** ()  
 Shutdown down this singleton.

bool **isAlive** ()  
 Returns if this singleton is alive (true until its shutdown begins).

## Private Functions

**SearchFilesystemHandler** ()

### 10.1.277 Class `sh::search::SearchManager`

**class** `sh::search::SearchManager` : **public** `QObject`, **public** `sh::base::Singleton`  
Manager for searches in the filesystem.

## Public Functions

`std::shared_ptr<sh::search::Search>` **requestSearch** (`std::shared_ptr<const sh::filesystem::Eurl>` *eurl*, `QString` *searchconfig*, `std::shared_ptr<sh::filesystem::FilesystemNode>` *searchrootnode*)

Requests a search.

**void** **removeSearch** (`std::shared_ptr<sh::search::Search>` *search*)  
Removes a search.

`std::shared_ptr<sh::search::Search>` **findSearch** (`std::shared_ptr<const sh::filesystem::Eurl>` *eurl*)  
Returns the *Search* instance for an eurl for a search root node.

**void** **addFactory** (`int` *index*, `std::shared_ptr<SearchCriterionFactory>` *f*)  
Adds a search criterion factory.

`QList<std::shared_ptr<SearchCriterionFactory>>` **factories** ()  
Returns the sorted list of search criterion factories.

**~SearchManager** ()

**void** **doShutdown** ()  
Executes singleton shutdown.

**void** **shutdown** ()  
Shutdown down this singleton.

**bool** **isAlive** ()  
Returns if this singleton is alive (true until its shutdown begins).

## Public Static Attributes

**const** `int` **REGISTER\_SEARCHCRITERION\_INDEX\_CORE** = 1000000  
Base value for display indexes of core (i.e. very important) search criteria.  
Used in *SearchManager::addFactory*.

**const** `int` **REGISTER\_SEARCHCRITERION\_INDEX\_NORMAL** = 2000000  
Base value for display indexes of search criteria with normal importance.  
Used in *SearchManager::addFactory*.

**const** `int` **REGISTER\_SEARCHCRITERION\_INDEX\_EXOTIC** = 3000000  
Base value for display indexes of exotic search criteria.  
Used in *SearchManager::addFactory*.

## Private Functions

**SearchManager** ( )

## Private Members

QList<std::shared\_ptr<*sh::search::Search*>> **\_searches**

QList<std::weak\_ptr<*sh::search::Search*>> **\_weak\_searches**

QMap<int, std::shared\_ptr<*SearchCriterionFactory*>> **\_factories**

## 10.1.278 Class *sh::search::criteria::AbstractActionDrivenSearchCriterion*

**class** *sh::search::criteria::AbstractActionDrivenSearchCriterion* : **public** *sh::search::SearchCriterion*  
 Abstract search criterion filtering by something which is configured in a wizard-like way by an action execution.

This provides an easy yet powerful programming interface, e.g. for usage in scripting.

Subclassed by *sh::scripting::api::ApiSearchCriterion*

## Public Functions

**AbstractActionDrivenSearchCriterion** (std::shared\_ptr<*sh::search::SearchCriterionFactory*>  
*factory*)

bool **match** (*sh::filesystem::Operation* \**op*, std::shared\_ptr<**const** *sh::filesystem::Eurl*> *eurl*)  
 Checks if a given file (by eurl) matches this criterion. .

bool **match2** (*sh::filesystem::Operation* \**op*, std::shared\_ptr<**const** *sh::filesystem::Eurl*> *eurl*) = 0

QString **valuedescription** ( )  
 Returns a human readable text describing the current criterion shortly and precisely. .

QString **serialize** ( )  
 Serializes this criterion to a string. See also *SearchCriterion::deserialize*.

std::shared\_ptr<*sh::search::SearchCriterionFactory*> **factory** ( )  
 Returns the factory this criterion has created (provides some more metadata).

## Public Members

QStringList **searchspec**

## Public Static Functions

void **\_createEditor** (std::shared\_ptr<*sh::search::SearchCriterion*> *cfg*,  
*sh::ui::SearchPanelConfiguration* \**panelcfg*)

void **\_editorUpdateConfiguration** (*sh::ui::SearchPanelConfiguration* \**panelcfg*,  
 std::shared\_ptr<*sh::search::SearchCriterion*> *c*)

std::shared\_ptr<*sh::search::SearchCriterion*> **deserialize** (QString *s*)  
 Deserializes this criterion from a string. See also *SearchCriterion::serialize*.

## Friends

**friend class** ActionAbstractActionDrivenSearchCriterionConfigure

### 10.1.279 Class `sh::search::criteria::AbstractStringSearchCriterion`

**class** `sh::search::criteria::AbstractStringSearchCriterion` : **public** `sh::search::SearchCriterion`

Abstract search criterion filtering by some string value (providing different modes like exact, fuzzy, regexp, ...).

Subclassed by `sh::search::criteria::FileContentSearchCriterion`, `sh::search::criteria::FilenameSearchCriterion`

## Public Types

**enum** `Mode`

*Values:*

**enumerator** `Simple`

**enumerator** `Exact`

**enumerator** `RegExp`

**enumerator** `Fuzzy`

## Public Functions

**AbstractStringSearchCriterion** (`std::shared_ptr<sh::search::SearchCriterionFactory>` *factory*)

**bool** **match** (`sh::filesystem::Operation` \**op*, `std::shared_ptr<const sh::filesystem::Eurl>` *eurl*)  
Checks if a given file (by eurl) matches this criterion. .

**QString** **valuedescription** ()  
Returns a human readable text describing the current criterion shortly and precisely. .

**QString** **serialize** ()  
Serializes this criterion to a string. See also `SearchCriterion::deserialize`.

`std::shared_ptr<sh::search::SearchCriterionFactory>` **factory** ()  
Returns the factory this criterion has created (provides some more metadata).

## Public Members

**QString** **string**

*Mode* **mode** = *Mode::Simple*

## Public Static Functions

```
void _createEditor (std::shared_ptr<sh::search::SearchCriterion> cfg,
                   ui::SearchPanelConfiguration *panelcfg)
void _editorUpdateConfiguration (sh::ui::SearchPanelConfiguration *panelcfg,
                                  std::shared_ptr<sh::search::SearchCriterion> c)
int levenshteinSubstringDistance (QString needle, QString haystack)
double relativeLevenshteinPartsSubstringDistance (QString needle, QString haystack)
std::shared_ptr<sh::search::SearchCriterion> deserialize (QString s)
    Deserializes this criterion from a string. See also SearchCriterion::serialize.
```

## Public Static Attributes

```
const QStringList ModeCodes
```

### 10.1.280 Class sh::search::criteria::ActionAbstractActionDrivenSearchCriterionConfigure

```
class sh::search::criteria::ActionAbstractActionDrivenSearchCriterionConfigure : public sh::act
```

## Public Functions

```
ActionAbstractActionDrivenSearchCriterionConfigure (std::shared_ptr<AbstractActionDrivenSearchCriterion>
                                                    cfg, QMutex *mutex)
void action (sh::actions::ActionExecutionInfo *info)
    The action implementation, i.e. what the actions should actually do.
void execute ()
    Executes this action.
void execute (sh::actions::ActionExecutionInfo *info)
    Executes this action.
QKeySequence shortcuthint ()
    Returns the keyboard shortcut for triggering this action.
bool shortcuthintTriggersOnCurrentDirectory ()
    Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry
    selection).
void setShortcuthint (QKeySequence shortcut, bool triggersOnCurrentDirectory = false)
    Sets the keyboard shortcut for triggering this action.
QString text ()
    Returns the displayed text for this action.
QString icon ()
    Returns the icon for this action.
bool enabled ()
    Checks if this action is enabled.
bool isChecked ()
    Checks if this action is checked (has a cross in the ui).
```

bool **isCheckable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<AbstractActionItem> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<AbstractActionItem> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>

> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **changed** ()

Emits when some data changed.

## Private Members

std::shared\_ptr<AbstractActionDrivenSearchCriterion> **\_cfg**  
 QMutex \*\_**mutex**

### 10.1.281 Class sh::search::criteria::ExtendedAttributeSearchCriterion

**class** *sh::search::criteria::ExtendedAttributeSearchCriterion* : **public** *sh::search::SearchCriterion*  
*Search* criterion filtering by extended attributes.

## Public Types

**enum** **Mode**  
*Values:*  
 enumerator **None**  
 enumerator **Simple**  
 enumerator **RegExp**  
 enumerator **Fuzzy**

## Public Functions

**ExtendedAttributeSearchCriterion** (std::shared\_ptr<*sh::search::SearchCriterionFactory*>  
*factory*)  
**bool** **match** (*sh::filesystem::Operation* \*op, std::shared\_ptr<**const** *sh::filesystem::Eurl*> eurl)  
 Checks if a given file (by eurl) matches this criterion. .  
**QString** **valuedescription** ()  
 Returns a human readable text describing the current criterion shortly and precisely. .  
**QString** **serialize** ()  
 Serializes this criterion to a string. See also *SearchCriterion::deserialize*.  
 std::shared\_ptr<*sh::search::SearchCriterionFactory*> **factory** ()  
 Returns the factory this criterion has created (provides some more metadata).

## Public Members

*Mode* **keymode** = *Mode::None*  
*Mode* **valuemode** = *Mode::Simple*  
**QString** **key**  
**QString** **value**

### Public Static Functions

```
void createEditor (std::shared_ptr<sh::search::SearchCriterion>                cfg,  
                  sh::ui::SearchPanelConfiguration *panelcfg)  
void editorUpdateConfiguration (sh::ui::SearchPanelConfiguration          *panelcfg,  
                                std::shared_ptr<sh::search::SearchCriterion> c)  
void doInitialize ()  
void doShutdown ()  
std::shared_ptr<sh::search::SearchCriterion> deserialize (QString s)  
    Deserializes this criterion from a string. See also SearchCriterion::serialize.
```

### Public Static Attributes

```
const QStringList ModeCodes
```

## 10.1.282 Class sh::search::criteria::FileContentSearchCriterion

```
class sh::search::criteria::FileContentSearchCriterion : public sh::search::criteria::AbstractStringSearchC  
    Search criterion filtering by file contents.
```

### Public Types

```
enum Mode  
    Values:  
    enumerator Simple  
    enumerator Exact  
    enumerator RegExp  
    enumerator Fuzzy
```

### Public Functions

```
FileContentSearchCriterion (std::shared_ptr<sh::search::SearchCriterionFactory> factory)  
bool match (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl)  
    Checks if a given file (by eurl) matches this criterion. .  
QString valuedescription ()  
    Returns a human readable text describing the current criterion shortly and precisely. .  
QString serialize ()  
    Serializes this criterion to a string. See also SearchCriterion::deserialize.  
std::shared_ptr<sh::search::SearchCriterionFactory> factory ()  
    Returns the factory this criterion has created (provides some more metadata).
```



## Public Members

QString **string**

*Mode* **mode** = *Mode::Simple*

## Public Static Functions

void **createEditor** (std::shared\_ptr<*sh::search::SearchCriterion*> *cfg*,  
*sh::ui::SearchPanelConfiguration* \*panelcfg)

void **editorUpdateConfiguration** (*sh::ui::SearchPanelConfiguration* \*panelcfg,  
std::shared\_ptr<*sh::search::SearchCriterion*> c)

void **doInitialize** ()

void **doShutdown** ()

void **\_createEditor** (std::shared\_ptr<*sh::search::SearchCriterion*> *cfg*,  
*ui::SearchPanelConfiguration* \*panelcfg)

void **\_editorUpdateConfiguration** (*sh::ui::SearchPanelConfiguration* \*panelcfg,  
std::shared\_ptr<*sh::search::SearchCriterion*> c)

int **levenshteinSubstringDistance** (QString *needle*, QString *haystack*)

double **relativeLevenshteinPartsSubstringDistance** (QString *needle*, QString *haystack*)

std::shared\_ptr<*sh::search::SearchCriterion*> **deserialize** (QString s)  
Deserializes this criterion from a string. See also *SearchCriterion::serialize*.

## Public Static Attributes

const QStringList **ModeCodes**

### 10.1.283 Class *sh::search::criteria::FilenameSearchCriterion*

**class** *sh::search::criteria::FilenameSearchCriterion* : **public** *sh::search::criteria::AbstractStringSearchCriterion*  
*Search* criterion filtering by file names.

## Public Types

**enum** **Mode**

*Values:*

**enumerator** **Simple**

**enumerator** **Exact**

**enumerator** **RegExp**

**enumerator** **Fuzzy**

## Public Functions

**FilenameSearchCriterion** (std::shared\_ptr<sh::search::SearchCriterionFactory> factory)

bool **match** (sh::filesystem::Operation \*op, std::shared\_ptr<const sh::filesystem::Eurl> eurl)  
Checks if a given file (by eurl) matches this criterion. .

QString **valuedescription** ()  
Returns a human readable text describing the current criterion shortly and precisely. .

QString **serialize** ()  
Serializes this criterion to a string. See also *SearchCriterion::deserialize*.

std::shared\_ptr<sh::search::SearchCriterionFactory> **factory** ()  
Returns the factory this criterion has created (provides some more metadata).

## Public Members

QString **string**

Mode **mode** = Mode::Simple

## Public Static Functions

void **createEditor** (std::shared\_ptr<sh::search::SearchCriterion> *cfg*,  
sh::ui::SearchPanelConfiguration \*panelcfg)

void **editorUpdateConfiguration** (sh::ui::SearchPanelConfiguration \*panelcfg,  
std::shared\_ptr<sh::search::SearchCriterion> c)

void **doInitialize** ()

void **doShutdown** ()

void **\_createEditor** (std::shared\_ptr<sh::search::SearchCriterion> *cfg*,  
ui::SearchPanelConfiguration \*panelcfg)

void **\_editorUpdateConfiguration** (sh::ui::SearchPanelConfiguration \*panelcfg,  
std::shared\_ptr<sh::search::SearchCriterion> c)

int **levenshteinSubstringDistance** (QString *needle*, QString *haystack*)

double **relativeLevenshteinPartsSubstringDistance** (QString *needle*, QString *haystack*)

std::shared\_ptr<sh::search::SearchCriterion> **deserialize** (QString *s*)  
Deserializes this criterion from a string. See also *SearchCriterion::serialize*.

## Public Static Attributes

const QStringList **ModeCodes**

### 10.1.284 Class `sh::search::criteria::MtimeSearchCriterion`

**class** `sh::search::criteria::MtimeSearchCriterion` : public `sh::search::SearchCriterion`  
*Search* criterion filtering by file modification times.

#### Public Functions

**MtimeSearchCriterion** (std::shared\_ptr<`sh::search::SearchCriterionFactory`> *factory*)

bool **match** (`sh::filesystem::Operation` \**op*, std::shared\_ptr<const `sh::filesystem::Eurl`> *eurl*)  
 Checks if a given file (by eurl) matches this criterion. .

QString **valuedescription** ()  
 Returns a human readable text describing the current criterion shortly and precisely. .

QString **serialize** ()  
 Serializes this criterion to a string. See also `SearchCriterion::deserialize`.

std::shared\_ptr<`sh::search::SearchCriterionFactory`> **factory** ()  
 Returns the factory this criterion has created (provides some more metadata).

#### Public Members

QDateTime **from**

QDateTime **to**

#### Public Static Functions

void **createEditor** (std::shared\_ptr<`sh::search::SearchCriterion`> *cfg*,  
                     `sh::ui::SearchPanelConfiguration` \**panelcfg*)

void **editorUpdateConfiguration** (`sh::ui::SearchPanelConfiguration` \**panelcfg*,  
                                     std::shared\_ptr<`sh::search::SearchCriterion`> *c*)

void **doInitialize** ()

void **doShutdown** ()

std::shared\_ptr<`sh::search::SearchCriterion`> **deserialize** (QString *s*)  
 Deserializes this criterion from a string. See also `SearchCriterion::serialize`.

### 10.1.285 Class `sh::settings::ProfileNode`

**class** `sh::settings::ProfileNode`  
 One entry in a Shallot settings profile, so one item as selectable in the ‘Manage settings’ dialog.

## Public Functions

**ProfileNode** ()  
Constructed only by the infrastructure and made available otherwise.

QString **nodeId** ()  
An identifier name.

void **setNodeId** (QString *nodeId*)  
Sets the identifier name.

std::shared\_ptr<const *filesystem::Eurl*> **eurl** ()  
The eurl of the directory, or 0 for global profile nodes.

void **setEurl** (std::shared\_ptr<const *filesystem::Eurl*> *eurl*)  
Sets the eurl.

QString **profilename** ()  
The profile name.

void **setProfilename** (QString *profilename*)  
Sets the profile name.

QStringList **inheritsFrom** ()  
List of profile names from which this node inherits.

void **setInheritsFrom** (QStringList *profilenames*)  
Sets list of profile names from which this node inherits.

bool **withSubfolders** ()  
If this node also applies recursively on subfolders.

void **setWithSubfolders** (bool *v*)  
Sets of this node also applies recursively on subfolders.

void **setSetting** (QString *name*, QString *value*, int *onlyInFileview* = -1)  
Stores a value for a setting for applying it later.

QList<QString> **getSettingNames** ()  
List of settings names which are set in this node.

QString **getSettingValue** (QString *name*)  
Gets the stored value of a setting by setting name.

int **getSettingOnlyInFileviewIndex** (QString *name*)  
Gets if a setting applies to a certain fileview or to the entire window.

## Private Members

QMap<QString, QString> **\_values**

QMap<QString, int> **\_valuesOnlyInFileviewIndex**

std::shared\_ptr<const *sh::filesystem::Eurl*> **\_eurl**

QString **\_profilename**

QStringList **\_inheritsFrom**

bool **\_withSubfolders**

QString **\_nodeId**

## 10.1.286 Class `sh::settings::Setting`

**class** `sh::settings::Setting`

Abstract base class for a Shallot setting.

Those have the greatest flexibility. They have to be stored manually by the user with many options. See Shallot documentation for details.

Use `sh::settings::SettingsRegistration` for registering an implementation.

Subclassed by `sh::scripting::api::ApiSetting`, `sh::settings::common::FileDetailsPanelVisible`,  
`sh::settings::common::FileViewIconListThumbDimension`, `sh::settings::common::FileViewPath`,  
`sh::settings::common::FileViewViewmode`, `sh::settings::common::JumpbarMode`,  
`sh::settings::common::NumberOfFilePanels`, `sh::settings::common::ShowHiddenFiles`,  
`sh::settings::common::ShowTree`, `sh::settings::common::SizeFormattingMode`,  
`sh::settings::common::StickyTreeview`, `sh::settings::common::WindowTitle`

### Public Functions

**Setting()**

Is (for subclasses) intended to be directly constructed from everywhere or by registering a factory somewhere.

**~Setting()**

QString **name()** = 0

Gets the internal name.

QString **description()** = 0

Gets the description text.

*SettingGroup* **group()** = 0

Gets the group;.

bool **isAdvancedSetting()** = 0

Is this an advanced setting?

void **setValue** (*sh::ui::FileView\**, QString)

Called from Shallot core when the value was set (for a per-fileview setting).

void **setValue** (QString)

Called from Shallot core when the value was set (for a not-per-fileview setting).

bool **isGlobal()** = 0

Does this setting apply globally or just for a certain subtree of nodes?

bool **isPerFileview()** = 0

Does this setting apply for each fileview individually or for the complete main window?

QString **getValue** (*sh::ui::FileView \*filelist*) = 0

Get the currently set value.

QString **valueDescription** (QString *value*)

Gets a human readable description text for a value.

## Public Static Functions

QString **getGroupDescription** (int g)

Low-level function which gets the description text of a group.

### 10.1.287 Class `sh::settings::SettingsManager`

**class** `sh::settings::SettingsManager` : **public** QObject, **public** `sh::base::Singleton`

Manager for Shallot settings.

This is the more flexible way of shallot customization. It includes everything you see in the ‘Manage saved settings’ or ‘Save settings’.

See `sh::settings::Setting` for more.

## Public Functions

void **applyPerEurlSettingsToFileView** (`sh::ui::FileView` \*list)

Applies all stored settings, which are per-eurl to a fileview.

void **applyGlobalSettingsToFileView** (`sh::ui::FileView` \*list)

Applies all stored settings, which are global to a fileview.

void **applyPerEurlSettingsToMainWindow** ()

Applies all stored settings, which are per-eurl to main window.

void **applyGlobalSettingsToMainWindow** ()

Applies all stored settings, which are global to a main window.

QList<`sh::settings::ProfileNode`> **getNodesForProfile** (QString profile)

Returns a list of all stored settings (as `ProfileNodes`) for a given profile.

`sh::settings::ProfileNode` \***getNodeById** (QString nodeId)

Searches and returns a `ProfileNode` by id.

std::shared\_ptr<`sh::settings::Setting`> **getSettingByName** (QString name)

Searches and returns a `Setting` by name.

QList<std::shared\_ptr<`sh::settings::Setting`>> **getAllSettings** ()

Returns a list of all available settings, primarily sorted by group.

QStringList **getProfileList** ()

Returns a list of all existing profiles.

void **removeProfile** (QString profile)

Removes a profile (including all `ProfileNodes` inside it).

void **removeNode** (QString nodeId)

Removes a `ProfileNode` by id.

void **storeProfile** (std::shared\_ptr<const `sh::filesystem::Eurl`> eurl, QString profilename, QMap<QString, QString> values, QMap<QString, int> onlyinfileviewindex, bool withSubfolders, QStringList inheritFrom)

Stores one setting into a profile.

**~SettingsManager** ()

void **doInitialize** ()

Executes singleton initialization.

void **doShutdown** ()  
 Executes singleton shutdown.

void **shutdown** ()  
 Shutdown down this singleton.

bool **isAlive** ()  
 Returns if this singleton is alive (true until its shutdown begins).

## Signals

void **profilesChanged** ()  
 Triggered when the list of profiles or the nodes inside it change.

Can also be triggered when nothing really changed!

## Private Functions

void **applyToFileView** (*sh::ui::FileView* \*list, QMap<QString, QString> settings)  
 Applies some settings (as string tuples) to a fileview.

void **applyToMainWindow** (QMap<QString, QString> settings)  
 Applies some settings (as string tuples) to main window.

QMap<QString, QString> **\_getSettings** (std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QString  
 profilename, int fileviewindex, bool directNode = true)  
 Computes all effective settings (including inheritance and sub-directories) for a given situation (global or  
 eurl-bound; fileview- or mainwindow-bound) and returns it as string tuples.

### Parameters

- eurl: The directory to consider as current (or nullptr for global settings).
- profilename: The current profile.
- fileviewindex: The fileview index (or -1 for mainwindow-bound settings).
- directNode: If the given directory is to be considered as the real current directory (not e.g. a parent).

QList<*SettingsFinderStructure*> **\_getSettings\_flat** (std::shared\_ptr<const  
*sh::filesystem::Eurl*> eurl, QString pro-  
 filename)

Used internally by *\_getSettings()*.

void **invalidateCache** ()  
 Invalidates the *\_getSettings* cache.

void **readStoredProfiles** ()  
 Reads all profiles (including all setting nodes) from filesystem.

**SettingsManager** ()

### Private Members

```
QMap<QString, QMap<QString, QString>> _getSettings_cache
QMap<QString, std::shared_ptr<sh::settings::Setting>> _settings
QMap<std::shared_ptr<const sh::filesystem::Eurl>, QMap<QString, QList<sh::settings::ProfileNode*>>> _profileNodes
QMap<QString, sh::settings::ProfileNode*> _profileNodesById
QList<QString> _profileNames
```

### Friends

```
friend class sh::settings::SettingsRegistration

class _MyHandler : public QXmlDefaultHandler
    Used internally for reading profile node XMLs.
```

### Public Functions

```
_MyHandler (sh::settings::ProfileNode *node, QString filename)

bool startElement (const QString &namespaceURI, const QString &localName, const
                  QString &qName, const QXmlAttributes &atts)

bool endElement (const QString &namespaceURI, const QString &localName, const
                QString &qName)
```

### Private Functions

```
void throwError ()
```

### Private Members

```
bool _stateStarted = true
bool _stateInProfile = false
sh::settings::ProfileNode *_node
QString _filename

class _SettingsFinderStructure
    Used internally by _getSettings().
```

### Public Functions

```
_SettingsFinderStructure (QMap<QString, QString> settings, QMap<QString, int> only-
                          infileviewindexes, QStringList inheritFrom, bool withSubfold-
                          ers)

_SettingsFinderStructure ()
```



**Public Members**

```

 QMap<QString, QString> _settings
 QMap<QString, int> _onlyinfileviewindexes
 QStringList _inheritFrom
 bool _withSubfolders
 int _onlyFileviewIndex

```

**10.1.288 Class sh::settings::SettingsManager::\_MyHandler**

**class** *sh::settings::SettingsManager::\_MyHandler* : **public** QDomDefaultHandler  
 Used internally for reading profile node XMLs.

**Public Functions**

```

 _MyHandler (sh::settings::ProfileNode *node, QString filename)
 bool startElement (const QString &namespaceURI, const QString &localName, const QString
                   &qName, const QDomAttributes &atts)
 bool endElement (const QString &namespaceURI, const QString &localName, const QString
                  &qName)

```

**Private Functions**

```

 void throwError ()

```

**Private Members**

```

 bool _stateStarted = true
 bool _stateInProfile = false
 sh::settings::ProfileNode * _node
 QString _filename

```

**10.1.289 Class sh::settings::SettingsManager::\_SettingsFinderStructure**

**class** *sh::settings::SettingsManager::\_SettingsFinderStructure*  
 Used internally by *\_getSettings()*.

### Public Functions

```
_SettingsFinderStructure ( QMap<QString, QString> settings, QMap<QString, int> onlyinfileviewindexes, QStringList inheritFrom, bool withSubfolders )  
_SettingsFinderStructure ( )
```

### Public Members

```
QMap<QString, QString> _settings  
QMap<QString, int> _onlyinfileviewindexes  
QStringList _inheritFrom  
bool _withSubfolders  
int _onlyFileviewIndex
```

## 10.1.290 Class sh::settings::SettingsRegistration

**class** *sh::settings::SettingsRegistration*

Used for registering a setting instance.

Constructing a *SettingsRegistration* automatically registers a *sh::settings::Setting*, deleting a one automatically unregisters it.

### Public Functions

```
SettingsRegistration ( std::shared_ptr<sh::settings::Setting> setting )  
    Is intended to be directly constructed from everywhere.  
~SettingsRegistration ( )  
std::shared_ptr<sh::settings::Setting> setting ( )
```

### Private Members

```
std::shared_ptr<sh::settings::Setting> _setting
```

## 10.1.291 Class sh::settings::common::FileDetailsPanelVisible

**class** *sh::settings::common::FileDetailsPanelVisible* : public *sh::settings::Setting*

## Public Functions

**FileDetailsPanelVisible** ()

QString **name** ()

Gets the internal name.

QString **description** ()

Gets the description text.

*sh::settings::SettingGroup* **group** ()

Gets the group;.

bool **isAdvancedSetting** ()

Is this an advanced setting?

void **setValue** (QString *value*)

Called from Shallot core when the value was set (for a not-per-fileview setting).

QString **getValue** (*sh::ui::FileView \*filelist*)

Get the currently set value.

bool **isGlobal** ()

Does this setting apply globally or just for a certain subtree of nodes?

bool **isPerFileview** ()

Does this setting apply for each fileview individually or for the complete main window?

QString **valueDescription** (QString *value*)

Gets a human readable description text for a value.

void **setValue** (*sh::ui::FileView\**, QString)

Called from Shallot core when the value was set (for a per-fileview setting).

## Public Static Functions

QString **getGroupDescription** (int *g*)

Low-level function which gets the description text of a group.

### 10.1.292 Class *sh::settings::common::FileViewIconListThumbDimension*

```
class sh::settings::common::FileViewIconListThumbDimension : public sh::settings::Setting
```

## Public Functions

**FileViewIconListThumbDimension** ()

QString **name** ()

Gets the internal name.

QString **description** ()

Gets the description text.

*sh::settings::SettingGroup* **group** ()

Gets the group;.

bool **isAdvancedSetting** ()

Is this an advanced setting?

void **setValue** (*sh::ui::FileView \*filelist*, QString *value*)  
Called from Shallot core when the value was set (for a per-fileview setting).

QString **getValue** (*sh::ui::FileView \*filelist*)  
Get the currently set value.

bool **isGlobal** ()  
Does this setting apply globally or just for a certain subtree of nodes?

bool **isPerFileview** ()  
Does this setting apply for each fileview individually or for the complete main window?

QString **valueDescription** (QString *value*)  
Gets a human readable description text for a value.

void **setValue** (QString)  
Called from Shallot core when the value was set (for a not-per-fileview setting).

### Public Static Functions

QString **getGroupDescription** (int *g*)  
Low-level function which gets the description text of a group.

## 10.1.293 Class *sh::settings::common::FileViewPath*

```
class sh::settings::common::FileViewPath: public sh::settings::Setting
```

### Public Functions

**FileViewPath** ()

QString **name** ()  
Gets the internal name.

QString **description** ()  
Gets the description text.

*sh::settings::SettingGroup* **group** ()  
Gets the group;.

bool **isAdvancedSetting** ()  
Is this an advanced setting?

void **setValue** (*sh::ui::FileView \*fileview*, QString *value*)  
Called from Shallot core when the value was set (for a per-fileview setting).

QString **getValue** (*sh::ui::FileView \*filelist*)  
Get the currently set value.

bool **isGlobal** ()  
Does this setting apply globally or just for a certain subtree of nodes?

bool **isPerFileview** ()  
Does this setting apply for each fileview individually or for the complete main window?

void **setValue** (QString)  
Called from Shallot core when the value was set (for a not-per-fileview setting).

QString **valueDescription** (QString *value*)  
 Gets a human readable description text for a value.

### Public Static Functions

QString **getGroupDescription** (int *g*)  
 Low-level function which gets the description text of a group.

## 10.1.294 Class `sh::settings::common::FileViewViewmode`

**class** `sh::settings::common::FileViewViewmode` : **public** `sh::settings::Setting`

### Public Functions

**FileViewViewmode** ()

QString **name** ()  
 Gets the internal name.

QString **description** ()  
 Gets the description text.

`sh::settings::SettingGroup` **group** ()  
 Gets the group;.

bool **isAdvancedSetting** ()  
 Is this an advanced setting?

void **setValue** (`sh::ui::FileView` \**fileview*, QString *value*)  
 Called from Shallot core when the value was set (for a per-fileview setting).

QString **getValue** (`sh::ui::FileView` \**filelist*)  
 Get the currently set value.

bool **isGlobal** ()  
 Does this setting apply globally or just for a certain subtree of nodes?

bool **isPerFileview** ()  
 Does this setting apply for each fileview individually or for the complete main window?

QString **valueDescription** (QString *value*)  
 Gets a human readable description text for a value.

void **setValue** (QString)  
 Called from Shallot core when the value was set (for a not-per-fileview setting).

### Public Static Functions

QString **getGroupDescription** (int *g*)  
 Low-level function which gets the description text of a group.

### 10.1.295 Class `sh::settings::common::JumpbarMode`

```
class sh::settings::common::JumpbarMode : public sh::settings::Setting
```

#### Public Functions

**JumpbarMode** ()

QString **name** ()

Gets the internal name.

QString **description** ()

Gets the description text.

*sh::settings::SettingGroup* **group** ()

Gets the group;.

bool **isAdvancedSetting** ()

Is this an advanced setting?

void **setValue** (QString *value*)

Called from Shallot core when the value was set (for a not-per-fileview setting).

QString **getValue** (*sh::ui::FileView \*filelist*)

Get the currently set value.

bool **isGlobal** ()

Does this setting apply globally or just for a certain subtree of nodes?

bool **isPerFileview** ()

Does this setting apply for each fileview individually or for the complete main window?

QString **valueDescription** (QString *value*)

Gets a human readable description text for a value.

void **setValue** (*sh::ui::FileView\**, QString)

Called from Shallot core when the value was set (for a per-fileview setting).

#### Public Static Functions

QString **getGroupDescription** (int *g*)

Low-level function which gets the description text of a group.

### 10.1.296 Class `sh::settings::common::NumberOfFilePanels`

```
class sh::settings::common::NumberOfFilePanels : public sh::settings::Setting
```

## Public Functions

**NumberOfFilePanels** ()

QString **name** ()

Gets the internal name.

QString **description** ()

Gets the description text.

*sh::settings::SettingGroup* **group** ()

Gets the group;.

bool **isAdvancedSetting** ()

Is this an advanced setting?

void **setValue** (QString *value*)

Called from Shallot core when the value was set (for a not-per-fileview setting).

QString **getValue** (*sh::ui::FileView \*filelist*)

Get the currently set value.

bool **isGlobal** ()

Does this setting apply globally or just for a certain subtree of nodes?

bool **isPerFileview** ()

Does this setting apply for each fileview individually or for the complete main window?

QString **valueDescription** (QString *value*)

Gets a human readable description text for a value.

void **setValue** (*sh::ui::FileView\**, QString)

Called from Shallot core when the value was set (for a per-fileview setting).

## Public Static Functions

QString **getGroupDescription** (int *g*)

Low-level function which gets the description text of a group.

### 10.1.297 Class *sh::settings::common::ShowHiddenFiles*

```
class sh::settings::common::ShowHiddenFiles : public sh::settings::Setting
```

## Public Functions

**ShowHiddenFiles** ()

QString **name** ()

Gets the internal name.

QString **description** ()

Gets the description text.

*sh::settings::SettingGroup* **group** ()

Gets the group;.

bool **isAdvancedSetting** ()

Is this an advanced setting?

void **setValue** (*sh::ui::FileView \*filelist*, QString *value*)  
Called from Shallot core when the value was set (for a per-fileview setting).

QString **getValue** (*sh::ui::FileView \*filelist*)  
Get the currently set value.

bool **isGlobal** ()  
Does this setting apply globally or just for a certain subtree of nodes?

bool **isPerFileview** ()  
Does this setting apply for each fileview individually or for the complete main window?

QString **valueDescription** (QString *value*)  
Gets a human readable description text for a value.

void **setValue** (QString)  
Called from Shallot core when the value was set (for a not-per-fileview setting).

### Public Static Functions

QString **getGroupDescription** (int *g*)  
Low-level function which gets the description text of a group.

## 10.1.298 Class *sh::settings::common::ShowTree*

```
class sh::settings::common::ShowTree : public sh::settings::Setting
```

### Public Functions

**ShowTree** ()

QString **name** ()  
Gets the internal name.

QString **description** ()  
Gets the description text.

*sh::settings::SettingGroup* **group** ()  
Gets the group;.

bool **isAdvancedSetting** ()  
Is this an advanced setting?

void **setValue** (QString *value*)  
Called from Shallot core when the value was set (for a not-per-fileview setting).

QString **getValue** (*sh::ui::FileView \*filelist*)  
Get the currently set value.

bool **isGlobal** ()  
Does this setting apply globally or just for a certain subtree of nodes?

bool **isPerFileview** ()  
Does this setting apply for each fileview individually or for the complete main window?

QString **valueDescription** (QString *value*)  
Gets a human readable description text for a value.



void **setValue** (*sh::ui::FileView\**, QString)  
 Called from Shallot core when the value was set (for a per-fileview setting).

### Public Static Functions

QString **getGroupDescription** (int *g*)  
 Low-level function which gets the description text of a group.

## 10.1.299 Class *sh::settings::common::SizeFormattingMode*

**class** *sh::settings::common::SizeFormattingMode* : **public** *sh::settings::Setting*

### Public Functions

**SizeFormattingMode** ()

QString **name** ()  
 Gets the internal name.

QString **description** ()  
 Gets the description text.

*sh::settings::SettingGroup* **group** ()  
 Gets the group;.

bool **isAdvancedSetting** ()  
 Is this an advanced setting?

void **setValue** (*sh::ui::FileView \*filelist*, QString *value*)  
 Called from Shallot core when the value was set (for a per-fileview setting).

QString **getValue** (*sh::ui::FileView \*filelist*)  
 Get the currently set value.

bool **isGlobal** ()  
 Does this setting apply globally or just for a certain subtree of nodes?

bool **isPerFileview** ()  
 Does this setting apply for each fileview individually or for the complete main window?

QString **valueDescription** (QString *value*)  
 Gets a human readable description text for a value.

void **setValue** (QString)  
 Called from Shallot core when the value was set (for a not-per-fileview setting).

### Public Static Functions

QString **getGroupDescription** (int *g*)  
 Low-level function which gets the description text of a group.

### 10.1.300 Class `sh::settings::common::StickyTreeview`

```
class sh::settings::common::StickyTreeview : public sh::settings::Setting
```

#### Public Functions

**StickyTreeview** ()

QString **name** ()

Gets the internal name.

QString **description** ()

Gets the description text.

*sh::settings::SettingGroup* **group** ()

Gets the group;.

bool **isAdvancedSetting** ()

Is this an advanced setting?

void **setValue** (QString *value*)

Called from Shallot core when the value was set (for a not-per-fileview setting).

QString **getValue** (*sh::ui::FileView \*filelist*)

Get the currently set value.

bool **isGlobal** ()

Does this setting apply globally or just for a certain subtree of nodes?

bool **isPerFileview** ()

Does this setting apply for each fileview individually or for the complete main window?

QString **valueDescription** (QString *value*)

Gets a human readable description text for a value.

void **setValue** (*sh::ui::FileView\**, QString)

Called from Shallot core when the value was set (for a per-fileview setting).

#### Public Static Functions

QString **getGroupDescription** (int *g*)

Low-level function which gets the description text of a group.

### 10.1.301 Class `sh::settings::common::WindowTitle`

```
class sh::settings::common::WindowTitle : public sh::settings::Setting
```

## Public Functions

**WindowTitle** ()

QString **name** ()

Gets the internal name.

QString **description** ()

Gets the description text.

*sh::settings::SettingGroup* **group** ()

Gets the group;.

bool **isAdvancedSetting** ()

Is this an advanced setting?

void **setValue** (QString *value*)

Called from Shallot core when the value was set (for a not-per-fileview setting).

QString **getValue** (*sh::ui::FileView \*filelist*)

Get the currently set value.

bool **isGlobal** ()

Does this setting apply globally or just for a certain subtree of nodes?

bool **isPerFileview** ()

Does this setting apply for each fileview individually or for the complete main window?

void **setValue** (*sh::ui::FileView\**, QString)

Called from Shallot core when the value was set (for a per-fileview setting).

QString **valueDescription** (QString *value*)

Gets a human readable description text for a value.

## Public Static Functions

QString **getGroupDescription** (int *g*)

Low-level function which gets the description text of a group.

### 10.1.302 Class *sh::tools::AtomicCounter*

**class** *sh::tools::AtomicCounter*

Internal tool for reference counting.

## Public Functions

**AtomicCounter** ()

int **value** ()

std::shared\_ptr<*AtomicCounter::Increment*> **increment** ()

int **doIncValue** ()

int **doDecValue** ()

**Private Members**

```
int _value = 0
QMutex _mutex
class Increment
```

**Public Functions**

```
Increment (AtomicCounter *ac)
~Increment ()
```

**Private Members**

```
AtomicCounter *_ac
```

**Friends**

```
friend class AtomicCounter
```

**10.1.303 Class sh::tools::AtomicCounter::Increment**

```
class sh::tools::AtomicCounter::Increment
```

**Public Functions**

```
Increment (AtomicCounter *ac)
~Increment ()
```

**Private Members**

```
AtomicCounter *_ac
```

**Friends**

```
friend class AtomicCounter
```

**10.1.304 Class sh::tools::Benchmarking**

```
class sh::tools::Benchmarking : public QObject, public sh::base::Singleton
    Tools for performance measurements.
```

## Public Types

**typedef** QPair<QString, qint64> **BenchmarkFrame**  
A pair of a benchmark keyname and a duration value.

## Public Functions

**Benchmarking** ()

void **shutdown** ()  
Shutdown down this singleton.

bool **isAlive** ()  
Returns if this singleton is alive (true until its shutdown begins).

### 10.1.305 Class `sh::tools::Bookmark`

**class** `sh::tools::Bookmark`  
A bookmark.

## Public Functions

**Bookmark** (QString *id*, QStringList *folder*, QString *label*, std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*, QString *tags*)  
Constructed only by the infrastructure and made available otherwise.

QString **id** ()  
The bookmark id (used in some methods of *sh::tools::BookmarkManager*).

QList<QString> **folder** ()  
The bookmark folder (the user interface calls it ‘Collections’).

QString **label** ()  
The bookmark label.

std::shared\_ptr<const *sh::filesystem::Eurl*> **eurl** ()  
The location this bookmark points to.

QString **tags** ()  
Internal information for bookkeeping.  
Can be used by external code for keeping track of dynamically created bookmarks.

## Private Members

QString **\_id**

QStringList **\_folder**

QString **\_label**

std::shared\_ptr<const *sh::filesystem::Eurl*> **\_eurl**

QString **\_tags**

### 10.1.306 Class `sh::tools::BookmarkManager`

**class** `sh::tools::BookmarkManager` : public `QObject`, public `sh::base::Singleton`

The bookmark manager.

Use it for getting or modifying bookmarks.

#### Public Functions

`QList<std::shared_ptr<sh::tools::Bookmark>>` **getBookmarks** ()

Returns a list of all stored bookmarks.

`bool` **hasBookmarks** ()

Checks if any bookmarks exist.

`QString` **addBookmark** (`QList<QString>` *folder*, `QString` *label*, `std::shared_ptr<const sh::filesystem::Eurl>` *eurl*, `QString` *tags* = `QString()`)

Adds a new bookmark.

**Return** The id of the new bookmark.

`void` **removeBookmark** (`QString` *id*)

Removes a bookmarks.

`void` **changeBookmark** (`QString` *id*, `QString` *label*, `std::shared_ptr<const sh::filesystem::Eurl>` *eurl*)

Changes data of a bookmark.

`void` **changeBookmarkTags** (`QString` *id*, `QString` *tags*)

Changes the tags of a bookmark.

`void` **moveBookmarkUp** (`QString` *id*)

Moves a bookmark up within its folder.

`void` **moveBookmarkDown** (`QString` *id*)

Moves a bookmark down within its folder.

`void` **moveBookmarkToFolder** (`QString` *id*, `QList<QString>` *folder*)

Changes to *folder* of a bookmark.

`void` **doInitialize** ()

Executes singleton initialization.

`void` **doShutdown** ()

Executes singleton shutdown.

`void` **shutdown** ()

Shutdown down this singleton.

`bool` **isAlive** ()

Returns if this singleton is alive (true until its shutdown begins).

## Signals

void **changed** ()

## Private Functions

**BookmarkManager** ()

void **moveBookmark** (QString *id*, int *direction*)

void **\_changeBookmark** (QString *id*, std::function<QString> std::shared\_ptr<Bookmark>  
> *label*, std::function<std::shared\_ptr<const sh::filesystem::Eurl>std::shared\_ptr<Bookmark>>  
*eurl*, std::function<QList<QString>std::shared\_ptr<Bookmark>> *folder*,  
std::function<QStringstd::shared\_ptr<Bookmark>> *tags*)

void **readBookmarks** ()

void **writeBookmarks** ()

## Private Members

std::shared\_ptr<sh::configuration::ConfigurationValue> **cfgvalBookmarks**

QList<std::shared\_ptr<Bookmark>> **\_bookmarks**

QMutex **\_bookmarksmutex**

### 10.1.307 Class sh::tools::DataExchange

**class** *sh::tools::DataExchange* : **public** QObject, **public** *sh::base::Singleton*  
Clipboard and DnD related tools.

## Public Types

**enum DataExchangeType**

A kind of data exchange movement.

*Values:*

**enumerator COPY**

**enumerator MOVE**

## Public Functions

bool **containsFileEntries** (const QMimeData\*)

Checks if a QMimeData contains any file entries we understand.

QMimeData \***getMimeDataFromFilesystemNodes** (QList<std::shared\_ptr<sh::filesystem::FilesystemNode>>,  
*DataExchangeType type*)

Returns a new QMimeData for a list of filesystem nodes and an exchange type.

std::shared\_ptr<sh::actions::AbstractActionItem> **createCopyAction** (const QMimeData \**src*,  
std::shared\_ptr<const  
*sh::filesystem::Eurl*> *dest*)

Returns a copy action for a source by QMimeData and a destination eurl.

```
std::shared_ptr<sh::actions::AbstractActionItem> createMoveAction (const  QMimeData  *src,
                                                                    std::shared_ptr<const
                                                                    sh::filesystem::Eurl> dest)
    Returns a move action for a source by QMimeData and a destination eurl.

std::shared_ptr<sh::actions::AbstractActionItem> createPasteAction (const  QMimeData  *src,
                                                                    std::shared_ptr<const
                                                                    sh::filesystem::Eurl> dest)
    Returns a paste action for a source by QMimeData and a destination eurl (copy or move by QMimeData).

QList<std::shared_ptr<const sh::filesystem::Eurl>> getSources (const  QMimeData *src, bool *is-
                                                                    Cut = 0)
    Returns a list of eurls (and if it is a cut/move exchange) from the QMimeData.

void doInitialize ()
    Executes singleton initialization.

void doShutdown ()
    Executes singleton shutdown.

void shutdown ()
    Shutdown down this singleton.

bool isAlive ()
    Returns if this singleton is alive (true until its shutdown begins).
```

### Public Static Attributes

```
QString FilelistTypeShallot = "x-special/shallot-copied-files"
QString FilelistTypeGnome = "x-special/gnome-copied-files"
QString FilelistTypeUrulist = "text/uri-list"
QString FilelistTypePlaintext = "text/plain"
```

### Private Functions

```
DataExchange ()
```

### Private Static Attributes

```
QString linebreak = "\n"
```

## 10.1.308 Class sh::tools::HistoryTracker

```
template<class T>
class sh::tools::HistoryTracker
    A data structure for tracking a history of data.
```

It is a templated container It is typically used for tracking the directory history (used by *sh::actions::common::ActionHistoryNavigate*, et al).



**Public Functions**

```

HistoryTracker ()
void visitValue (T v)
int count ()
QList<HistoryEntry> forwardList ()
QList<HistoryEntry> backwardList ()
void revisitValue (int idx)

```

**Public Members**

```

const int HISTORY_SIZE = 10

```

**Private Members**

```

QList<T> _items
int _current = -1
class HistoryEntry

```

**Public Functions**

```

HistoryEntry (int i, T v)
int index ()
T value ()

```

**Private Members**

```

int _index
T _value

```

**10.1.309 Class sh::tools::HistoryTracker::HistoryEntry**

```

class sh::tools::HistoryTracker::HistoryEntry

```

**Public Functions**

```

HistoryEntry (int i, T v)
int index ()
T value ()

```

### Private Members

int **\_index**

T **\_value**

## 10.1.310 Class sh::tools::Jsonable

**class** *sh::tools::Jsonable*

An interface for objects which can return a json representation as QJsonObject.

Subclassed by *sh::ui::web::WebDialog*, *sh::ui::web::WebFilePropertyDialog::WebFilePropertyDialogTabActionsView*, *sh::ui::web::WebFilePropertyDialog::WebFilePropertyDialogTabIconTextBannerView*, *sh::ui::web::WebFilePropertyDialog::WebFilePropertyDialogTabTableView*, *sh::ui::web::WebFilePropertyDialog::WebFileProp*

### Public Functions

QJsonObject **toJson** () = 0

Returns the json respresentation as QJsonObject.

## 10.1.311 Class sh::tools::LocalFile

**class** *sh::tools::LocalFile* : **public** QFile

A QFile (and a QIODevice) for local file access.

It also stores the information whether this is a temporary location or the permanent one.

### Public Functions

**LocalFile** (QString &*file*, bool *isTemp* = false)

bool **isTemp** ()

### Private Members

bool **\_istemp**

## 10.1.312 Class sh::tools::LocalFilesystemWatcher

**class** *sh::tools::LocalFilesystemWatcher* : **public** QObject, **public** *sh::base::Singleton*

Used for watching parts of the local filesystem.

Depending on the compile flags and your system, this functionality might not be available. If so, the methods are no-ops.

## Public Functions

qint64 **addFile** (QString *path*)

Watches one more file.

See also *removeFile()*.

void **removeFile** (qint64 *id*)

Stops watching a file (by the return value of *addFile()*).

qint64 **addDirectory** (QString *path*)

Watches one more directory.

See also *removeDirectory()*.

void **removeDirectory** (qint64 *id*)

Stops watching a directory (by the return value of *addDirectory()*).

**~LocalFilesystemWatcher** ()

void **doInitialize** ()

Executes singleton initialization.

void **doShutdown** ()

Executes singleton shutdown.

void **shutdown** ()

Shutdown down this singleton.

bool **isAlive** ()

Returns if this singleton is alive (true until its shutdown begins).

## Signals

void **elementModified** (QString *name*, QList<qint64> *ids*)

void **elementDeleted** (QString *name*, QList<qint64> *ids*)

void **elementCreated** (QString *name*, QList<qint64> *ids*)

## Private Functions

**LocalFilesystemWatcher** ()

void **emitElementModified** (QString *name*, QList<qint64> *ids*)

void **emitElementDeleted** (QString *name*, QList<qint64> *ids*)

void **emitElementCreated** (QString *name*, QList<qint64> *ids*)

### Private Members

```
InotifyThread *inotifyThread
QMutex inotifymutex
QHash<int, QList<qint64>> inotifywd2ids
QHash<qint64, int> id2inotifywd
QHash<qint64, QString> id2path
qint64 nextEid = 0
QMutex inotifythreadmutex
QWaitCondition inotifythreadmutexwait
```

### Friends

```
friend class InotifyThread
class InotifyThread : public QThread
```

### Public Functions

```
InotifyThread (LocalFilesystemWatcher *watcher)
```

### Friends

```
friend class LocalFilesystemWatcher
```

## 10.1.313 Class `sh::tools::LocalFilesystemWatcherConnector`

```
class sh::tools::LocalFilesystemWatcherConnector : public QObject, public sh::base::Singleton
    Observes the list of file views via VisibleViews and controls LocalFilesystemWatcher with that for observing the
    filesystem for changes.
```

### Public Functions

```
void doInitialize ()
    Executes singleton initialization.

void doShutdown ()
    Executes singleton shutdown.

void shutdown ()
    Shutdown down this singleton.

bool isAlive ()
    Returns if this singleton is alive (true until its shutdown begins).
```

## Private Functions

**LocalFilesystemWatcherConnector** ()

QList<QPair<std::shared\_ptr<*sh::filesystem::FilesystemNode*>, *sh::filesystemhandlers::LocalFilesystemHandler*\*>> **getAllLocalFilesystemWatchers**

## Private Members

QHash<std::shared\_ptr<const *sh::filesystem::Eurl*>, QList<qint64>> **dir2watcherids**

QHash<qint64, std::shared\_ptr<const *sh::filesystem::Eurl*>> **watcherid2dir**

### 10.1.314 Class *sh::tools::LocalFilesystemWatcher::InotifyThread*

**class** *sh::tools::LocalFilesystemWatcher::InotifyThread* : **public** QThread

## Public Functions

**InotifyThread** (*LocalFilesystemWatcher* \**watcher*)

## Friends

**friend class** LocalFilesystemWatcher

### 10.1.315 Class *sh::tools::Misc*

**class** *sh::tools::Misc*

## Public Static Functions

QByteArray **iconToPngByteArray** (QIcon *icon*, int *sizeInPt*)  
Computes a png representation for an icon.

QByteArray **iconToBase64SrcEncoding** (QIcon *icon*, int *sizeInPt*)  
Computes a html-compatible ‘data:image/png;base64,...’ png representation for an icon.

QByteArray **hash** (QStringList *data*)  
Hashes string data in a cryptographical way (with a salt).  
Use with *compareHash()*.

QByteArray **hashUnsalted** (QStringList *data*)  
Hashes string data in a cryptographical way without a salt.  
Use with *compareHash()* or just compare strings.

bool **compareHash** (QByteArray *hash*, QStringList *data*)  
Check if a hash is matching to given string data.

QByteArray **qjsonToJson** (QJsonObject *o*)  
Returns a json string for a QJsonObject, QJsonValue or QJsonArray.

QByteArray **qjsonToJson** (QJsonValue *o*)

QByteArray **qjsonToJson** (QJsonArray *a*)

QByteArray **qmapToJson** (QVariantMap *m*)  
Returns a json string for a QVariantMap.

QByteArray **jsonableToJson** (*Jsonable* \**a*)  
Returns a json string from a *Jsonable*.

QByteArray **randomBytes** (int *length* = 32)  
Returns a random byte array.

QByteArray **generateUniqueHash** ()  
Returns a (mostly) random generated unique hash.

void **makeHttpRequest** (QUrl *url*, QByteArray \**responseBody* = nullptr, QString \**mimeType* =  
nullptr, int \**httpStatus* = nullptr)  
Makes an http request (as a client).  
*httpStatus* can return 0 for network errors and similar.

## Private Functions

**Misc** ()

## Private Static Functions

QByteArray **\_hash** (QStringList *data*, bool *salted*, QByteArray *\_withSalt*)

## Private Static Attributes

QByteArray **\_jundefined**

QNetworkAccessManager **\_qnetwork**

## 10.1.316 Class sh::tools::OperationsCache

**class** *sh::tools::OperationsCache* : **public** QObject, **public** *sh::base::Singleton*  
A cached factory for readonly *sh::filesystem::Operation* instances.

You can use it for fetching an *sh::filesystem::Operation* for a certain *sh::filesystem::Eurl*. Two restrictions apply:

- Data might be slightly outdated.
- Just for read-only access.

## Public Functions

std::shared\_ptr<*sh::filesystem::Operation*> **getOperationForContainer** (std::shared\_ptr<const  
*sh::filesystem::Eurl*>  
*container*)

Returns a cached *sh::filesystem::Operation* for a container (by eurl).

void **doInitialize** ()  
Executes singleton initialization.

void **doShutdown** ()  
Executes singleton shutdown.

void **shutdown** ()  
 Shutdown down this singleton.

bool **isAlive** ()  
 Returns if this singleton is alive (true until its shutdown begins).

### Private Functions

**OperationsCache** ()

### Private Members

QHash<std::shared\_ptr<const *sh::filesystem::Eurl*>, std::shared\_ptr<*sh::filesystem::Operation*>> **cache**

QMutex **cachemutex**

## 10.1.317 Class *sh::tools::ReadDataDevice*

**class** *sh::tools::ReadDataDevice* : **public** QIODevice  
 A QIODevice implementation, acting as an abstract base class for other subclasses.

The provided interface is not as generic but easier than the original one for many easier tasks.

Subclassed by *sh::scripting::api::ApiReadDataDevice*

### Public Functions

**ReadDataDevice** ()

QByteArray **getdata** () = 0  
 This method returns the next available chunk of data. It may return empty arrays whenever temporarily no data is available. Returning a null array (with `QByteArray::isNull () == true`) marks the end of the stream. .

### Private Members

QByteArray **current**

int **currentlen** = 0

int **currentconsumed** = 0

## 10.1.318 Class *sh::tools::ThumbnailManager*

**class** *sh::tools::ThumbnailManager* : **public** QObject, **public** *sh::base::Singleton*  
 Creates thumbnail images for filesystem nodes.

This is a cached source.

## Public Functions

void **requestThumbnail** (std::shared\_ptr<sh::filesystem::FilesystemNode> node, int width, int height, std::function<void> QIcon  
> callback = 0) Requests a thumbnail in a given size for a given node.

### Parameters

- **callback**: Called when the thumbnail is ready.

bool **getThumbnail** (std::shared\_ptr<sh::filesystem::FilesystemNode> node, int width, int height, QIcon \*result, bool \*outdated = 0, bool \*refreshRequested = 0)  
Returns a thumbnail in a given size for a given node from the current cache.

void **invalidateCache** ()  
Invalidates the thumbnail cache.

void **addThumbnailProvider** (int index, std::shared\_ptr<ThumbnailProvider> provider)  
Adds a thumbnail provider.

void **doInitialize** ()  
Executes singleton initialization.

void **shutdown** ()  
Shutdown down this singleton.

bool **isAlive** ()  
Returns if this singleton is alive (true until its shutdown begins).

## Signals

void **thumbnailAvailable** (std::shared\_ptr<sh::filesystem::FilesystemNode> node)  
Emitted when a new thumbnail is available.

## Public Static Attributes

const int **REGISTER\_THUMBNAILPROVIDER\_INDEX\_CORE** = 1000000  
Base value for display indexes of core (i.e. very important) thumbnail providers.  
Used in ThumbnailProvider::addThumbnailProvider.

const int **REGISTER\_THUMBNAILPROVIDER\_INDEX\_NORMAL** = 2000000  
Base value for display indexes of thumbnail providers with normal importance.  
Used in ThumbnailProvider::addThumbnailProvider.

const int **REGISTER\_THUMBNAILPROVIDER\_INDEX\_EXOTIC** = 3000000  
Base value for display indexes of exotic thumbnail providers.  
Used in ThumbnailProvider::addThumbnailProvider.

const int **REGISTER\_THUMBNAILPROVIDER\_INDEX\_FALLBACK** = 4000000  
Base value for display indexes of fallback thumbnail providers (those who try to get something when everything else failed).  
Used in ThumbnailProvider::addThumbnailProvider.



## Private Functions

```
ThumbnailManager ()
void worker ()
void _enforce_capacity ()
```

## Private Members

```
QHash<sh::filesystem::FilesystemNode*, Thumbnail*> thumbnails
QMutex thumbnailslock
QList<ThumbnailRequest> requestQueue
int maxworkers
int runningworkers
QMutex workermutex
int capacity
qint64 _curr_accessTime = 0
QHash<int, std::shared_ptr<ThumbnailProvider>>> _thumbnailProviderMap
QList<std::shared_ptr<ThumbnailProvider>>> _thumbnailProviders
QMutex _thumbnailProvidersMutex
class Thumbnail
```

## Public Functions

```
Thumbnail (QIcon icon, qint64 validUntil, qint64 accessTime,
           std::weak_ptr<sh::filesystem::FilesystemNode> node, int width, int height)
```

## Public Members

```
QIcon icon
qint64 validUntil
qint64 accessTime
bool refreshRequested
std::weak_ptr<sh::filesystem::FilesystemNode> node
int width
int height
class ThumbnailRequest
```

### Public Functions

```
ThumbnailRequest (std::weak_ptr<sh::filesystem::FileSystemNode> node, int width, int height,  
                  QList<std::function<void>()> QIcon  
                  >> callbacks  
bool operator== (ThumbnailRequest const &b)
```

### Public Members

```
std::weak_ptr<sh::filesystem::FileSystemNode> node  
int width  
int height  
QList<std::function<void (QIcon) >> callbacks  
class ThumbnailSortStruct
```

### Public Functions

```
ThumbnailSortStruct (sh::filesystem::FileSystemNode *node, qint64 accessTime)
```

### Public Members

```
sh::filesystem::FileSystemNode *node  
qint64 accessTime
```

## 10.1.319 Class sh::tools::ThumbnailManager::Thumbnail

```
class sh::tools::ThumbnailManager::Thumbnail
```

### Public Functions

```
Thumbnail (QIcon icon, qint64 validUntil, qint64 accessTime,  
           std::weak_ptr<sh::filesystem::FileSystemNode> node, int width, int height)
```

### Public Members

```
QIcon icon  
qint64 validUntil  
qint64 accessTime  
bool refreshRequested  
std::weak_ptr<sh::filesystem::FileSystemNode> node  
int width  
int height
```

### 10.1.320 Class `sh::tools::ThumbnailManager::ThumbnailRequest`

```
class sh::tools::ThumbnailManager::ThumbnailRequest
```

#### Public Functions

```
ThumbnailRequest (std::weak_ptr<sh::filesystem::FileSystemNode> node, int width, int height,
                  QList<std::function<void>() QIcon
                  >> callbacks
bool operator== (ThumbnailRequest const &b)
```

#### Public Members

```
std::weak_ptr<sh::filesystem::FileSystemNode> node
int width
int height
QList<std::function<void (QIcon) >> callbacks
```

### 10.1.321 Class `sh::tools::ThumbnailManager::ThumbnailSortStruct`

```
class sh::tools::ThumbnailManager::ThumbnailSortStruct
```

#### Public Functions

```
ThumbnailSortStruct (sh::filesystem::FileSystemNode *node, qint64 accessTime)
```

#### Public Members

```
sh::filesystem::FileSystemNode *node
qint64 accessTime
```

### 10.1.322 Class `sh::tools::ThumbnailProvider`

```
class sh::tools::ThumbnailProvider
```

Subclassed by *sh::scripting::api::ApiThumbnailProvider*, *sh::tools::thumbnailproviders::DefaultImageThumbnailProvider*, *sh::tools::thumbnailproviders::FfmpegVideoThumbnailProvider*, *sh::tools::thumbnailproviders::ImageMagickPdfThumbnailProvider*, *sh::tools::thumbnailproviders::PlaintextThumbnailProvider*

## Public Functions

void **getThumbnail** (*sh::filesystem::Operation* \*operation, std::shared\_ptr<*sh::filesystem::FilesystemNode*> node, QString contentType, int width, int height, QIcon \*icon) = 0

### 10.1.323 Class *sh::tools::UserDirLock*

**class** *sh::tools::UserDirLock*

Locks the Shallot user directory during exclusive usage.

This lock is rather heavy-weight and should be used only if necessary!

## Public Functions

**UserDirLock** ()

Is intended to be directly constructed from everywhere.

**~UserDirLock** ()

## Private Members

bool **downlock**

QString **slockpref**

QString **slockfile**

## Private Static Attributes

QThread \***currentThread** = 0

QMutex **mutex**

### 10.1.324 Class *sh::tools::VisibleViews*

**class** *sh::tools::VisibleViews* : public *sh::base::Singleton*

Used for keeping track of which fileviews show which directories.

Whenever the view goes to other directories, this manager gets notified and redirects the notification to some other parts of the program.

## Public Functions

std::shared\_ptr<QObject> **viewEnteredDirectory** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> dir)

Called when a view enters the given directory node. .

void **registerOnEnteredHandlers** (std::function<void> std::shared\_ptr<*sh::filesystem::FilesystemNode*> dir  
> handlerRegisters a function to be called whenever the view just entered the given directory node.

void **registerOnLeftHandlers** (std::function<void> std::shared\_ptr<*sh::filesystem::FilesystemNode*> dir  
> handlerRegisters a function to be called whenever the view just left the given directory node.

```

void doInitialize ()
    Executes singleton initialization.

void doShutdown ()
    Executes singleton shutdown.

void shutdown ()
    Shutdown down this singleton.

bool isAlive ()
    Returns if this singleton is alive (true until its shutdown begins).

```

### Private Functions

```
VisibleViews ()
```

### Private Members

```

QHash<std::shared_ptr<sh::filesystem::FilesystemNode>, int> visibleDirs

QList<std::function<void (std::shared_ptr<sh::filesystem::FilesystemNode>) >> onenteredhandlers

QList<std::function<void (std::shared_ptr<sh::filesystem::FilesystemNode>) >> onlefthandlers

```

## 10.1.325 Class sh::tools::accounts::AbstractAccountsProvider

```
class sh::tools::accounts::AbstractAccountsProvider
```

Abstract base class for accounts provider.

Implement this class (and register an instance of it) in order to add support for something like a password/account manager.

Subclassed by *sh::tools::accounts::FallbackAccountsProvider*, *sh::tools::accounts::LibsecretAccountsProvider*

### Public Functions

```
AbstractAccountsProvider ()
```

```
~AbstractAccountsProvider ()
```

```
void findAccounts (std::shared_ptr<Account> pattern, QList<std::shared_ptr<Account>> &result) =
    0
    Finds account info by a given pattern.
```

```
bool storeAccount (std::shared_ptr<Account> account) = 0
    Stores account infos.
```

### 10.1.326 Class `sh::tools::accounts::Account`

**class** `sh::tools::accounts::Account`  
*Account* data (for accessing network drives).

#### Public Functions

**Account** () = default

**Account** (const *Account* &*other*) = default

#### Public Members

QString **username**

QString **domain**

QString **server**

int **serverport** = -1

QString **path**

QString **protocol**

QString **authtype**

QByteArray **authinfo**

### 10.1.327 Class `sh::tools::accounts::AccountsManager`

**class** `sh::tools::accounts::AccountsManager` : public `sh::base::Singleton`  
Storage for account data (for accessing network drives).

#### Public Functions

QList<std::shared\_ptr<*Account*>> **findAccounts** (std::shared\_ptr<*Account*> *pattern*)  
Finds account infos by a given pattern.

void **storeAccount** (std::shared\_ptr<*Account*> *account*)  
Stores account infos.

void **registerProvider** (std::shared\_ptr<*AbstractAccountsProvider*> *provider*)  
Registers a new accounts provider.

void **doInitialize** ()  
Executes singleton initialization.

void **doShutdown** ()  
Executes singleton shutdown.

void **shutdown** ()  
Shutdown down this singleton.

bool **isAlive** ()  
Returns if this singleton is alive (true until its shutdown begins).

## Private Functions

**AccountsManager** ( )

## Private Members

QList<std::shared\_ptr<*AbstractAccountsProvider*>> **accountsproviders**

QMutex **mutex**

### 10.1.328 Class sh::tools::accounts::FallbackAccountsProvider

**class** *sh::tools::accounts::FallbackAccountsProvider* : **public** *sh::tools::accounts::AbstractAccountsProvider*

A fallback accounts provider which at least stores user names (i.e. no passwords) locally on disk.

## Public Functions

**FallbackAccountsProvider** ( )

void **findAccounts** (std::shared\_ptr<*Account*> *pattern*, QList<std::shared\_ptr<*Account*>> &*result*)

Finds account info by a given pattern.

bool **storeAccount** (std::shared\_ptr<*Account*> *account*)

Stores account infos.

## Public Static Functions

void **doInitialize** ( )

void **doShutdown** ( )

## Private Functions

QList<std::shared\_ptr<*Account*>> **\_find\_rem\_account\_helper** (std::shared\_ptr<*Account*> *pattern*, bool *remove*)

## Private Members

QString **accountsdir**

### 10.1.329 Class sh::tools::accounts::LibsecretAccountsProvider

**class** *sh::tools::accounts::LibsecretAccountsProvider* : **public** *sh::tools::accounts::AbstractAccountsProvider*

An accounts provider based on gnome-keyring.

## Public Functions

**LibsecretAccountsProvider** ()

void **findAccounts** (std::shared\_ptr<*Account*> *pattern*, QList<std::shared\_ptr<*Account*>> &*result*)  
Finds account info by a given pattern.

bool **storeAccount** (std::shared\_ptr<*Account*> *account*)  
Stores account infos.

## Public Static Functions

void **doInitialize** ()

void **doShutdown** ()

## Private Members

QByteArray **sauthtype** = QString("authtype").toUtf8()

QByteArray **sdomain** = QString("domain").toUtf8()

QByteArray **spath** = QString("object").toUtf8()

QByteArray **sprotocol** = QString("protocol").toUtf8()

QByteArray **sserver** = QString("server").toUtf8()

QByteArray **sserverport** = QString("port").toUtf8()

QByteArray **susername** = QString("user").toUtf8()

### 10.1.330 Class *sh::tools::filetypes::FileTypeManager*

**class** *sh::tools::filetypes::FileTypeManager* : **public** QObject, **public** *sh::base::Singleton*  
Utilities for dealing with file types.

It can determine the type of a file (png, plaintext, html, ...), it can provide information about how to open them with an external program, and more.

For most tasks it uses a pluggable interface. Actual implementations of strategies for those tasks reside in separate classes in this namespace.

## Public Functions

QString **determineMimetype** (*sh::filesystem::Operation* \**op*, std::shared\_ptr<**const** *sh::filesystem::Eurl*> *eurl*)  
Determines the mimetype for one file.

QHash<std::shared\_ptr<**const** *sh::filesystem::Eurl*>, QString> **determineMimetype** (*sh::filesystem::Operation* \**op*,  
QList<std::shared\_ptr<**const** *sh::filesystem::Eurl*>>  
*items*)  
Determines the mimetypes for a list of files.



`QList<std::shared_ptr<OpenMethod>> getOpenMethods (QString mimetype,  
 QList<std::shared_ptr<sh::filesystem::FilesystemNode>>  
nodes)`

Determines how to open a file with a given mimetype with an external program.

`QList<std::shared_ptr<OpenMethod>> getAllOpenMethods ()`

Returns a list of all known infos how to open a file external programs.

This has roughly one entry for each installed program on the user's system, which can graphically open files.

`void doInitialize ()`

Executes singleton initialization.

`void doShutdown ()`

Executes singleton shutdown.

`void shutdown ()`

Shutdown down this singleton.

`bool isAlive ()`

Returns if this singleton is alive (true until its shutdown begins).

## Private Functions

`FileTypeManager (QObject *parent = 0)`

## Private Members

`QList<std::shared_ptr<MimetypeDeterminationStrategy>> _mimetypeDeterminationMethods`

`QList<std::shared_ptr<OpenMethodDeterminationStrategy>> _openMethodDeterminationMethods`

`QList<std::shared_ptr<MimetypeInformationRetrievalStrategy>> _mimetypeInformationRetrievalMethods`

`QMutex _mutex`

`class MimetypeDeterminationStrategy`

Abstract base class for a mimetype determination strategy.

Subclassed by `sh::tools::filetypes::FreedesktopOrgToolsMimetypeDeterminationStrategy`,

`sh::tools::filetypes::SuffixListMimetypeDeterminationStrategy`, `sh::tools::filetypes::UnixFileToolMimetypeDeterminationStrategy`

## Public Functions

`QHash<std::shared_ptr<const sh::filesystem::Eurl>, QString> determineMimetype (sh::filesystem::Operation  
*op,  
 QList<std::shared_ptr<const  
sh::filesystem::Eurl>>  
items) =  
 0`

`~MimetypeDeterminationStrategy ()`

`class MimetypeInformationRetrievalStrategy`

Abstract base class for mimetype information retrieval strategy.

Subclassed by `sh::tools::filetypes::FreeDesktopOrgMimetypeInformationRetrievalStrategy`

### Public Functions

```
QStringList getMimeTypeSubclasses (QString mimetype)  
~MimeTypeInformationRetrievalStrategy ()  
  
class OpenMethodDeterminationStrategy  
    Abstract base class for a 'open method' determination strategy.  
  
    Subclassed by sh::tools::filetypes::FreedesktopOrgToolsOpenMethodDeterminationStrategy,  
    sh::tools::filetypes::UserDefinedOpenMethodDeterminationStrategy
```

### Public Functions

```
QList<std::shared_ptr<OpenMethod>> getOpenMethods (QString mimetype,  
    QList<std::shared_ptr<sh::filesystem::FilesystemNode>>  
    nodes) = 0  
  
~OpenMethodDeterminationStrategy ()
```

## 10.1.331 Class *sh::tools::filetypes::FileTypeManager::MimeTypeDeterminationStrategy*

```
class sh::tools::filetypes::FileTypeManager::MimeTypeDeterminationStrategy  
    Abstract base class for a mimetype determination strategy.  
  
    Subclassed by sh::tools::filetypes::FreedesktopOrgToolsMimeTypeDeterminationStrategy,  
    sh::tools::filetypes::SuffixListMimeTypeDeterminationStrategy, sh::tools::filetypes::UnixFileToolMimeTypeDeterminationStrategy
```

### Public Functions

```
QHash<std::shared_ptr<const sh::filesystem::Eurl>, QString> determineMimeType (sh::filesystem::Operation  
    *op,  
    QList<std::shared_ptr<const sh::filesystem::Eurl>>  
    items) = 0  
  
~MimeTypeDeterminationStrategy ()
```

## 10.1.332 Class *sh::tools::filetypes::FileTypeManager::MimeTypeInformationRetrievalStrategy*

```
class sh::tools::filetypes::FileTypeManager::MimeTypeInformationRetrievalStrategy  
    Abstract base class for mimetype information retrieval strategy.  
  
    Subclassed by sh::tools::filetypes::FreeDesktopOrgMimeTypeInformationRetrievalStrategy
```

## Public Functions

QStringList **getMimeTypeSubclasses** (QString *mimetype*)  
**~MimeTypeInformationRetrievalStrategy** ()

### 10.1.333 Class `sh::tools::filetypes::FileTypeManager::OpenMethodDeterminationStrategy`

**class** `sh::tools::filetypes::FileTypeManager::OpenMethodDeterminationStrategy`

Abstract base class for a ‘open method’ determination strategy.

Subclassed by `sh::tools::filetypes::FreedesktopOrgToolsOpenMethodDeterminationStrategy`,  
`sh::tools::filetypes::UserDefinedOpenMethodDeterminationStrategy`

## Public Functions

QList<std::shared\_ptr<OpenMethod>> **getOpenMethods** (QString *mimetype*,  
 QList<std::shared\_ptr<sh::filesystem::FilesystemNode>>  
*nodes*) = 0  
**~OpenMethodDeterminationStrategy** ()

### 10.1.334 Class `sh::tools::filetypes::FreeDesktopOrgMimeTypeInformationRetrievalStrategy`

**class** `sh::tools::filetypes::FreeDesktopOrgMimeTypeInformationRetrievalStrategy` : public `sh::tools::filetypes::OpenMethodDeterminationStrategy`

Tries to determine some mimetype information with the freedesktop.org specs.

## Public Functions

**FreeDesktopOrgMimeTypeInformationRetrievalStrategy** ()  
 QStringList **getMimeTypeSubclasses** (QString *mimetype*)

## Private Members

QHash<QString, QStringList> **\_mimeSubclassOf**  
 QMutex **\_mutex**  
 QWaitCondition **\_cond\_initd**  
 bool **\_initd** = false  
**class MimeTypeInfo** : public QXmlDefaultHandler

### Public Functions

**MimetypeInfo** () = default

bool **startElement** (const QString &namespaceURI, const QString &localName, const QString &qName, const QDomAttributes &atts)

bool **endElement** (const QString &namespaceURI, const QString &localName, const QString &qName)

### Public Members

QStringList **subClassOf**

## 10.1.335 Class sh::tools::filetypes::FreeDesktopOrgMimetypeInformationRetrievalStrategy::MimetypeIn

```
class sh::tools::filetypes::FreeDesktopOrgMimetypeInformationRetrievalStrategy : MimetypeIn
```

### Public Functions

**MimetypeInfo** () = default

bool **startElement** (const QString &namespaceURI, const QString &localName, const QString &qName, const QDomAttributes &atts)

bool **endElement** (const QString &namespaceURI, const QString &localName, const QString &qName)

### Public Members

QStringList **subClassOf**

## 10.1.336 Class sh::tools::filetypes::FreedesktopOrgToolsMimetypeDeterminationStrategy

```
class sh::tools::filetypes::FreedesktopOrgToolsMimetypeDeterminationStrategy : public sh::tools::
    Tries to determine a file's mimetype with the freedesktop.org tools.
```

### Public Functions

**FreedesktopOrgToolsMimetypeDeterminationStrategy** (sh::tools::filetypes::FileTypeManager \*manager)

QHash<std::shared\_ptr<const sh::filesystem::Eurl>, QString> **determineMimetype** (sh::filesystem::Operation \*op, QList<std::shared\_ptr<const sh::filesystem::Eurl>> items)

## Private Members

QMutex **\_mutex**  
 QString **\_pathToFileTool**  
 const QRegularExpression **\_reMimetype**

## Private Static Attributes

std::shared\_ptr<sh::configuration::ConfigurationValue> **cfgvalXdgmimePath** = sh::configuration::ConfigurationManager::in

## 10.1.337 Class sh::tools::filetypes::FreedesktopOrgToolsOpenMethodDeterminationStrategy

**class** sh::tools::filetypes::FreedesktopOrgToolsOpenMethodDeterminationStrategy : public sh::tools::OpenMethodDeterminationStrategy  
 Tries to determine a ‘open method’ for a file with the freedesktop.org tools.

## Public Functions

**FreedesktopOrgToolsOpenMethodDeterminationStrategy** ()  
 ~**FreedesktopOrgToolsOpenMethodDeterminationStrategy** ()  
 QList<std::shared\_ptr<sh::tools::filetypes::OpenMethod>> **getOpenMethods** (QString *mimetype*,  
 QList<std::shared\_ptr<sh::filesystem::FilesystemNode>> *nodes*)

## Private Functions

QString **\_parseValue** (QStringList *content*, QString *key*)  
 void **\_parseExecLine** (QString *exeline*, QString \**command*, QStringList \**arguments*)  
 QString **\_escapeExecLineToken** (QString *s*)

## Private Members

QMultiMap<QString, ApplicationEntry\*> **\_mimetype2applicationEntry**  
 QMutex **\_mutex**  
 QWaitCondition **\_cond\_initied**  
 bool **\_initied** = false  
 struct **ApplicationEntry**

### Public Members

QString **name**  
QStringList **mimetypes**  
QString **command**  
QStringList **commandargs**  
QIcon **icon**  
bool **hidden**

## 10.1.338 Class `sh::tools::filetypes::SuffixListMimetypeDeterminationStrategy`

**class** `sh::tools::filetypes::SuffixListMimetypeDeterminationStrategy` : **public** `sh::tools::filetypes::File`  
Tries to determine a file's mimetype with an internal lookup table of file extensions.

### Public Functions

**SuffixListMimetypeDeterminationStrategy** ()  
QHash<std::shared\_ptr<const `sh::filesystem::Eurl`>, QString> **determineMimetype** (`sh::filesystem::Operation`  
\**op*,  
QList<std::shared\_ptr<const `sh::filesystem::Eurl`>>  
*items*)

### Private Members

QHash<QString, QString> **\_suffixToMimetype**  
QMutex **\_mutex**

## 10.1.339 Class `sh::tools::filetypes::UnixFileToolMimetypeDeterminationStrategy`

**class** `sh::tools::filetypes::UnixFileToolMimetypeDeterminationStrategy` : **public** `sh::tools::filetypes::File`  
Tries to determine a file's mimetype with the `unix file` tool.

### Public Functions

**UnixFileToolMimetypeDeterminationStrategy** (`sh::tools::filetypes::FileTypeManager`  
\**manager*)  
QHash<std::shared\_ptr<const `sh::filesystem::Eurl`>, QString> **determineMimetype** (`sh::filesystem::Operation`  
\**op*,  
QList<std::shared\_ptr<const `sh::filesystem::Eurl`>>  
*items*)

## Public Static Attributes

`std::shared_ptr<sh::configuration::ConfigurationValue> cfgvalFilePath = sh::configuration::ConfigurationManager::instance->getCfgvalFilePath();`

## Private Members

`QString _pathToFileTool`

`const QRegularExpression _reMimetype`

`QMutex _mutex`

### 10.1.340 Class `sh::tools::filetypes::UserDefinedOpenMethodDeterminationStrategy`

**class** `sh::tools::filetypes::UserDefinedOpenMethodDeterminationStrategy` : **public** `sh::tools::filetypes::OpenMethodDeterminationStrategy`

Tries to determine a ‘open method’ for a file by information the user stored before.

## Public Functions

`QList<std::shared_ptr<sh::tools::filetypes::OpenMethod>> getOpenMethods (QString mimetype, QList<std::shared_ptr<sh::filesystem::FilesystemNode>> nodes) override`

`void storeCustomOpenMethod (QList<std::shared_ptr<sh::filesystem::FilesystemNode>> nodes, QString mimetype, std::shared_ptr<sh::tools::filetypes::OpenMethod> m, bool rememberForMimetype, std::shared_ptr<const sh::filesystem::Eurl> rememberForDirectory, bool rememberForFile)`

Stores a custom open method.

`void doInitialize ()`  
Executes singleton initialization.

`void doShutdown ()`  
Executes singleton shutdown.

`void shutdown ()`  
Shutdown down this singleton.

`bool isAlive ()`  
Returns if this singleton is alive (true until its shutdown begins).

## Private Functions

`UserDefinedOpenMethodDeterminationStrategy ()`

### Private Members

QMutex **\_mutex**

## 10.1.341 Class `sh::tools::thumbnailproviders::DefaultImageThumbnailProvider`

**class** `sh::tools::thumbnailproviders::DefaultImageThumbnailProvider` : **public** `sh::tools::ThumbnailProvider`  
Thumbnail provider for images.

### Public Functions

**DefaultImageThumbnailProvider** () = default

void **getThumbnail** (`sh::filesystem::Operation` \*operation, std::shared\_ptr<`sh::filesystem::FilesystemNode`>  
node, QString contentType, int width, int height, QIcon \*icon) **override**

### Public Static Functions

void **doInitialize** ()

void **doShutdown** ()

## 10.1.342 Class `sh::tools::thumbnailproviders::FfmpegVideoThumbnailProvider`

**class** `sh::tools::thumbnailproviders::FfmpegVideoThumbnailProvider` : **public** QObject, **public** `sh::tools::ThumbnailProvider`  
Thumbnail provider for videos using the ffmpeg tool.

### Public Functions

**FfmpegVideoThumbnailProvider** ()

void **getThumbnail** (`sh::filesystem::Operation` \*operation, std::shared\_ptr<`sh::filesystem::FilesystemNode`>  
node, QString contentType, int width, int height, QIcon \*icon) **override**

### Public Static Functions

void **doInitialize** ()

void **doShutdown** ()

### Private Members

QString **pathToFfmpegTool**

QString **pathToFfprobeTool**

**const** QRegularExpression **reDuration**

QMutex **mutexReDuration**



### Private Static Attributes

`std::shared_ptr<sh::configuration::ConfigurationValue> cfgvalFfmpegPath = sh::configuration::ConfigurationManager::inst`

## 10.1.343 Class `sh::tools::thumbnailproviders::ImageMagickPdfThumbnailProvider`

**class** `sh::tools::thumbnailproviders::ImageMagickPdfThumbnailProvider` : **public** `QObject`, **public** `sh::ThumbnailProvider`  
Thumbnail provider for videos using the ffmpeg tool.

### Public Functions

**ImageMagickPdfThumbnailProvider** ()

void **getThumbnail** (*sh::filesystem::Operation* \*operation, std::shared\_ptr<*sh::filesystem::FilesystemNode*> node, *QString* contentType, int width, int height, *QIcon* \*icon) **override**

### Public Static Functions

void **doInitialize** ()

void **doShutdown** ()

### Private Members

*QString* **pathToImagemagickConvertTool**

### Private Static Attributes

`std::shared_ptr<sh::configuration::ConfigurationValue> cfgvalImagemagickConvertPath = sh::configuration::Configura`

## 10.1.344 Class `sh::tools::thumbnailproviders::PlaintextThumbnailProvider`

**class** `sh::tools::thumbnailproviders::PlaintextThumbnailProvider` : **public** `sh::tools::ThumbnailProvider`  
Thumbnail provider for plain text.

### Public Functions

**PlaintextThumbnailProvider** ()

void **getThumbnail** (*sh::filesystem::Operation* \*operation, std::shared\_ptr<*sh::filesystem::FilesystemNode*> node, *QString* contentType, int width, int height, *QIcon* \*icon) **override**

### Public Static Functions

void **doInitialize** ()

void **doShutdown** ()

### Private Members

QColor **brandingcolor**

## 10.1.345 Class `sh::ui::AboutDialog`

**class** `sh::ui::AboutDialog` : **public** `sh::ui::Dialog`

The ‘About’ dialog.

Subclassed by `sh::ui::qt::QtAboutDialog`, `sh::ui::web::WebAboutDialog`

### Public Functions

**AboutDialog** ()

qint64 **dialogId** ()

Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

bool **isInited** ()

Returns if this dialog is initialized.

Must be called in main thread.

bool **wasAccepted** ()

Returns if this dialog was ‘accepted’ by the user (typically by clicking some ‘OK’ button).

Must be called in main thread.

void **waitClosed** ()

Wait until the user closed the dialog in some way.

May be called in any thread.

void **close** ()

Closes the dialog.

Must be called in main thread.

bool **wasClosed** ()

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \***manager** ()

Returns the *DialogManager* which hosts this dialog.

### 10.1.346 Class `sh::ui::ActionExecutionInfoDialog`

**class** `sh::ui::ActionExecutionInfoDialog`: **public** `sh::actions::ActionExecutionUserFeedback`  
 Progress dialog for action executions.

Subclassed by `sh::ui::qt::QtActionExecutionInfoDialog`, `sh::ui::web::WebActionExecutionInfoDialog`

#### Public Types

**enum** `MessageBoxButton`

Buttons in a message box from *ActionExecutionUserFeedback*.

*Values:*

**enumerator** `NONE` = 0

**enumerator** `OK` = 1 << 0

**enumerator** `Continue` = 1 << 1

**enumerator** `Cancel` = 1 << 2

**enumerator** `Retry` = 1 << 3

**enumerator** `Yes` = 1 << 4

**enumerator** `No` = 1 << 5

#### Public Functions

**ActionExecutionInfoDialog** (`sh::actions::ActionExecutionInfo *info`)

void **setDetails** (QString *fv*, QString *fob*, QString *tv*, QString *tob*) = 0  
 Sets the item details text ('from a/foo.jpg', 'to b/foo.jpg'). .

void **setHead** (QString *txt*) = 0  
 Sets the header text. .

void **setProgress** (bool *isDeterminate*, quint64 *done*, quint64 *all*, QString *text*) = 0  
 Sets the progress. .

bool **isLogicallyVisible** ()  
 Returns if this dialog is logically visible (i.e. set visible by the action).

void **setLogicallyVisible** (bool *v*)  
 Sets if this dialog is logically visible (i.e. set visible by the action). .

bool **isBackground** ()  
 Returns if this dialog is currently in background mode (i.e. not visible).

void **setBackground** (bool *v*)  
 Sets if this dialog is currently in background mode (i.e. not visible). .

void **setForceForeground** (bool *v*)  
 Sets if this dialog is currently forced to be visible in foreground. .

int **messageBox** (QString *text*, QList<QString> *answers*, QString *icon* = QString(), int *defaultanswer* = -1, int *cancelanswer* = -1, QList<QString> *answericons* = QList<QString>()) = 0

int **inputBox** (QString *text*, QList<QString> *answers*, QString \**value*, QString *icon* = QString(), int *defaultanswer* = -1, int *cancelanswer* = -1, int *valuePreselectFrom* = -1, int *valuePreselectTo* = -1) = 0

```
int multilineInputDialog (QString text, QList<QString> answers, QString *value, QString icon =  
    QString(), int defaultanswer = -1, int cancelanswer = -1) = 0  
  
int simpleChooserGridform (QString text, GridformEntries *entries, QStringList answers, int de-  
    faultanswer = -1, int cancelanswer = -1) = 0  
  
bool credentialsDialog (QString text, bool showDomain, bool showUsername, bool showPass-  
    word, bool showAnonymous, bool showRemember, QString *domain,  
    QString *username, QString *password, bool *isAnonymous, bool *is-  
    Remember) = 0  
  
bool unixPermissionsDialog (bool *userMayRead, bool *userMayWrite, bool *userMayExecute,  
    bool *groupMayRead, bool *groupMayWrite, bool *groupMayEx-  
    ecute, bool *othersMayRead, bool *othersMayWrite, bool *others-  
    MayExecute, bool *sticky, bool *setuid, bool *setgid, QStringList  
    users, QStringList groups, QString *ownerUser, QString *owner-  
    Group) = 0  
  
QString simpleInputDialog (QString text, QString deflt, int valuePreselectFrom = -1, int valuePrese-  
    lectTo = -1)  
  
int simpleMessageBox (QString text, int buttons = (int)MessageBoxButton::OK, QString icon =  
    QString(), int defaultbutton = (MessageBoxButton)0, int cancelbutton =  
    (MessageBoxButton)0)
```

### 10.1.347 Class sh::ui::ActionExecutionInfoPanel

**class** *sh::ui::ActionExecutionInfoPanel*

Abstract class for a status bar info-panel for an action execution.

Subclassed by *sh::ui::qt::QtActionExecutionInfoPanel*, *sh::ui::web::WebActionExecutionInfoPanel*

#### Public Functions

```
void setLabel (QString s) = 0  
    Sets the label text. .  
  
void setProgress (bool isProgressDeterminate, quint64 progressDone, quint64 progressAll) = 0  
    Sets the progress. .  
  
void setForceForeground (bool v) = 0  
    Sets if the associated action is currently forced to be visible in foreground. .  
  
void setPanelVisible (bool v) = 0  
    Sets if the panel is visible. .  
  
void setWidth (int width) = 0  
    Sets the panel with (in pixels). .  
  
void onClicked (std::function<void>  
    > fcnQObject *owner = 0) Sets a handler for a click on the button (optionally bound to an owner lifetime).  
  
void onDestroyed (std::function<void>  
    > fcnQObject *owner = 0) Sets a handler for panel removal (optionally bound to an owner lifetime).  
  
void onVisibilityChanged (std::function<void>  
    > fcnQObject *owner = 0) Sets a handler for panel visibility changes (optionally bound to an owner lifetime).  
  
~ActionExecutionInfoPanel ()
```

### 10.1.348 Class `sh::ui::ColumnDimensions`

**class** `sh::ui::ColumnDimensions`

Auxiliary data structure for persistent column dimensions.

It stores the width of a column whenever it changes and restores it when needed (mostly when Shallot starts).

#### Public Functions

**ColumnDimensions** (int *index*)

Constructed only by the infrastructure and made available otherwise.

void **store** ()

bool **remove** ()

void **setWidth** (QString *col*, int *width*)

int **getWidth** (QString *col*, int *defaultval*)

#### Public Members

int **index**

#### Private Functions

std::unique\_ptr<QSettings> **db** ()

#### Private Members

QHash<QString, int> **dims**

### 10.1.349 Class `sh::ui::Dialog`

**class** `sh::ui::Dialog`: **public** std::enable\_shared\_from\_this<*Dialog*>

Abstract subclass for Shallot dialogs (in child windows).

Subclasses of *Dialog* represent particular dialog types (like *ManageBookmarksDialog*).

For each ui mode, there typically is an implementation for all of these types, implementing one of this dialog types subclasses (mentioned before) and also some ui mode specific class.

Note: Unless stated otherwise, all methods must be called from main thread!

Note: It's not allowed to show one web dialog instance more than once. After closing it, it's sole use is to fetch answer data from it.

Note: You should not create instances directly. It's required to use *DialogManager::createAndShowDialog()* for actually showing those dialogs. Even this should not be used directly, because call depends on the current ui mode. Use the show\*() methods in *sh::ui::MainWindow* for creating dialogs in a ui mode independent way.

Subclassed by *sh::ui::AboutDialog*, *sh::ui::ExceptionDialog*, *sh::ui::FilePropertyDialog*, *sh::ui::LogViewDialog*, *sh::ui::ManageBookmarksDialog*, *sh::ui::ManageProfilesDialog*, *sh::ui::OpenWithDialog*, *sh::ui::StoreProfileDialog*, *sh::ui::TuningDialog*

## Public Functions

**Dialog()**

**~Dialog()**

qint64 **dialogId()**

Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

bool **isInited()**

Returns if this dialog is initialized.

Must be called in main thread.

bool **wasAccepted()**

Returns if this dialog was 'accepted' by the user (typically by clicking some 'OK' button).

Must be called in main thread.

void **waitClosed()**

Wait until the user closed the dialog in some way.

May be called in any thread.

void **close()**

Closes the dialog.

Must be called in main thread.

bool **wasClosed()**

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \***manager()**

Returns the *DialogManager* which hosts this dialog.

## Private Functions

void **setup**(*DialogManager* \**manager*, qint64 *dialogId*)

Sets manager and id for this dialog (right after creation).

## Private Members

*DialogManager* \*\_**manager**

qint64 **\_dialogId** = -1

bool **\_closed** = false

bool **\_inited** = false

## Friends

```
friend class MainWindow
friend class DialogManager
```

### 10.1.350 Class `sh::ui::DialogManager`

**class** `sh::ui::DialogManager`

Creates and drives dialogs (in child windows), i.e. instances of *Dialog*.

Subclass it for implementing the low level mechanisms for handling dialogs in a particular ui mode (e.g. qt, web).

The infrastructure will use one of this implementations for creating and managing most kinds of dialogs.

Subclassed by `sh::ui::qt::QtDialogManager`, `sh::ui::web::WebDialogManager`

## Public Functions

`std::shared_ptr<Dialog> getDialogById (qint64 id)`

Returns a *Dialog* by dialog id.

Must be called in main thread.

`QList<std::shared_ptr<Dialog>> getAllDialogs ()`

Returns a list of all dialogs currently shown (in order of creation).

Must be called in main thread.

`QList<qint64> getAllDialogIds ()`

Returns a list of dialog ids of all dialogs currently shown (in order of creation).

Must be called in main thread.

`template<class TDlg, typename ...Args>`

`std::shared_ptr<TDlg> createAndShowDialog (Args... args)`

Creates and shows a *Dialog* by class and constructor parameters.

Those dialogs (TDlg) have to implement *Dialog* (typically a subclass of it, representing a particular dialog, like *FilePropertyDialog*), and also some ui mode (e.g. qt, web) specific class (depends on that ui mode).

You typically should not have to use it outside of *MainWindow* or subclasses. The *MainWindow* interface allows to create all available dialogs in a ui mode independent way.

May be called in any thread.

**Return** The created *Dialog* instance.

`~DialogManager ()`

### Private Functions

void **closeDialog** (std::shared\_ptr<*Dialog*> *dialog*)  
Closes this dialog, including internal bookkeeping.

Must be called in main thread.

void **initAndShowDialog** (std::shared\_ptr<*Dialog*> *dialog*)  
Initializes and shows a freshly created dialog.

Must be called in main thread.

bool **wasAccepted** (std::shared\_ptr<*Dialog*> *dialog*) = 0  
Returns if the dialog was ‘accepted’ by the user (typically by clicking some ‘OK’ button).

The result is undefined before the dialog was closed.

Must be called in main thread.

void **waitClosed** (std::shared\_ptr<*Dialog*> *dialog*) = 0  
Wait until the user closed the dialog in some way.

May be called in any thread.

void **stopDialog** (std::shared\_ptr<*Dialog*> *dialog*) = 0  
This method implements the ui mode specific mechanism for stopping (i.e. closing) a dialog.

Must be called in main thread.

void **showDialog** (std::shared\_ptr<*Dialog*> *dialog*) = 0  
This method implements the ui mode specific mechanism for showing a dialog.

Must be called in main thread.

### Private Members

QHash<qint64, std::shared\_ptr<*Dialog*>> **\_openDialogs**

qint64 **\_nextDialogId** = 0

### Friends

**friend class** Dialog

## 10.1.351 Class sh::ui::ExceptionDialog

**class** *sh::ui::ExceptionDialog* : **public** *sh::ui::Dialog*  
The ‘Exception’ dialog.

Subclassed by *sh::ui::qt::QtExceptionDialog*, *sh::ui::web::WebExceptionDialog*



## Public Functions

**ExceptionDialog** (QString *error1*, QString *error2*, QString *details*, QString *icon*, bool *mayRetry*, bool *mayClose*, bool *mayCancel*, bool *showLoglabelAndDetails*)

### Parameters

- **error1**: The 1st error text.
- **error2**: The 2nd error text.
- **details**: The error details.
- **icon**: An icon (as name resolveable by *sh::base::IconManager*).
- **mayRetry**: If it's allowed to retry.
- **mayClose**: If it's allowed to just close and continue without closing Shallot.
- **mayCancel**: If it's allowed to cancel, i.e. throwing a *sh::exceptions::CancelException*.
- **showLoglabelAndDetails**: If the dialog shall allow to read the details.

QString **error1** ()

Returns the 1st error text.

QString **error2** ()

Returns the 2nd error text.

QString **details** ()

Returns the error details.

QString **icon** ()

Returns the icon (as name resolveable by *sh::base::IconManager*).

bool **mayRetry** ()

Returns if it's allowed to retry.

bool **mayClose** ()

Returns if it's allowed to just close and continue without closing Shallot.

bool **mayCancel** ()

Returns if it's allowed to cancel, i.e. throwing a *sh::exceptions::CancelException*.

bool **showLoglabelAndDetails** ()

Returns if the dialog shall allow to read the details.

*ExceptionDialogResult* **answer** () = 0

Returns the answer the user has given in this dialog.

qint64 **dialogId** ()

Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

bool **isInitied** ()

Returns if this dialog is initialized.

Must be called in main thread.

bool **wasAccepted** ()

Returns if this dialog was 'accepted' by the user (typically by clicking some 'OK' button).

Must be called in main thread.

void **waitClosed** ()  
Wait until the user closed the dialog in some way.  
May be called in any thread.

void **close** ()  
Closes the dialog.  
Must be called in main thread.

bool **wasClosed** ()  
Returns if this dialog was closed.  
Must be called in main thread.

*DialogManager* \***manager** ()  
Returns the *DialogManager* which hosts this dialog.

### Private Members

QString **\_error1**  
QString **\_error2**  
QString **\_details**  
QString **\_icon**  
bool **\_mayRetry**  
bool **\_mayClose**  
bool **\_mayCancel**  
bool **\_showLoglabelAndDetails**

## 10.1.352 Class sh::ui::FilePropertyDialog

**class** *sh::ui::FilePropertyDialog* : **public** *sh::ui::Dialog*  
The ‘File Properties’ dialog.  
Subclassed by *sh::ui::qt::QtFilePropertyDialog*, *sh::ui::web::WebFilePropertyDialog*

### Public Functions

**~FilePropertyDialog** ()

**FilePropertyDialog** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> *nodes*)

void **refresh** () = 0  
Refreshes the dialog content.

QList<std::shared\_ptr<*FilePropertyDialogTab*>> **tabs** ()  
Returns a list of all tabs.

*sh::ui::FilePropertyDialogTabActionsView* \***widgetAt** (std::shared\_ptr<*FilePropertyDialogTab*> *tab*,  
int *i*)  
Returns the widget from a tab at a certain index.

int **widgetCount** (std::shared\_ptr<*FilePropertyDialogTab*> *tab*)  
Returns the number of widgets in a tab.

*sh::ui::FilePropertyDialogTabTableView* \***createTabViewTable** () = 0

Creates a new tab view table sub-widget.

*sh::ui::FilePropertyDialogTabTextView* \***createTabViewText** () = 0

Creates a new tab view text sub-widget.

*sh::ui::FilePropertyDialogTabIconTextBannerView* \***createTabViewIconTextBanner** () = 0

Creates a new tab view icon text banner sub-widget.

qint64 **dialogId** ()

Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

bool **isInited** ()

Returns if this dialog is initialized.

Must be called in main thread.

bool **wasAccepted** ()

Returns if this dialog was 'accepted' by the user (typically by clicking some 'OK' button).

Must be called in main thread.

void **waitClosed** ()

Wait until the user closed the dialog in some way.

May be called in any thread.

void **close** ()

Closes the dialog.

Must be called in main thread.

bool **wasClosed** ()

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \***manager** ()

Returns the *DialogManager* which hosts this dialog.

## Public Static Functions

void **addTabFactory** (int *i*, std::shared\_ptr<*FilePropertyDialogTabFactory*> *factory*)

Adds a *FilePropertyDialogTabFactory* so the Properties dialogs show its content.

See also the REGISTER\_FILEPROPERTYDIALOGTAB macro.

### Parameters

- *i*: An index which controls the display order. Use one of the REGISTER\_FILEPROPERTYDIALOGTAB\_INDEX\_\* values in *FilePropertyDialogTabFactory* as base values.

## Private Members

QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> **\_nodes**

QList<*TabViewStruct*> **\_tabs**

## Private Static Attributes

QHash<int, std::shared\_ptr<*FilePropertyDialogTabFactory*>> **\_propertytabs**

## Friends

**friend class** FilePropertyDialogTab

### 10.1.353 Class *sh::ui::FilePropertyDialogTab*

**class** *sh::ui::FilePropertyDialogTab* : **public** std::enable\_shared\_from\_this<*FilePropertyDialogTab*>  
Abstract base class for one tab in the Properties dialog (containing a list of widgets).

Subclass and register this class for providing additional content in the Properties dialog.

Register it by using the REGISTER\_FILEPROPERTYDIALOGTAB macro (or by adding a *FilePropertyDialogTabFactory* via *FilePropertyDialog::addTabFactory*).

Subclassed by *sh::filepropertydialogtabs::FilePropertyDialogTabExtendedAttributes*,  
*sh::filepropertydialogtabs::FilePropertyDialogTabGeneral*, *sh::filepropertydialogtabs::FilePropertyDialogTabUnixPermissions*,  
*sh::filepropertydialogtabs::FilePropertyDialogTabWindows*, *sh::scripting::api::ApiFilePropertyDialogTab*

## Public Functions

QString **title** () = 0  
The tab title. .

QString **titleWithoutMnemonic** ()  
The tab title without mnemonic.

QList<QString> **properties** () = 0  
Returns the list of property labels (one entry for each widget).

Each property is displayed with a header and a widget (created in *createWidget*) with some content (populated in *updateWidget*).

void **populateWidget** (int *i*, *FilePropertyDialogTabActionsView* \**widget*) = 0  
Populates the tab widget for the *i*-th property.

Typically uses the *FilePropertyDialog::create\** methods to create sub-widgets.

See also *dialog()*.

void **updateWidget** (int *i*, *FilePropertyDialogTabActionsView* \**widget*, *sh::filesystem::Operation* \**op*)  
= 0  
Populates the widget for the *i*-th property with actual content. .

void **refresh** ()  
Refreshes the complete content.

Typically called after some user actions in a property widget require that all the other content must be refreshed.

`QList<std::shared_ptr<sh::filesystem::FilesystemNode>> nodes ()`  
 The nodes to show.

`FilePropertyDialogTabActionsView *widgetAt (int i)`  
 Returns the widget for the i-th property.

`int widgetCount ()`  
 Returns the number of widgets.

`std::shared_ptr<FilePropertyDialog> dialog ()`  
 Returns the associated dialog.

`~FilePropertyDialogTab ()`

## Private Members

`std::weak_ptr<FilePropertyDialog> _dialog`

## Friends

`friend class FilePropertyDialog`

`friend class FilePropertyDialogTabFactoryByFunction`

`friend class sh::scripting::api::ApiFilePropertyDialogTabFactory`

### 10.1.354 Class sh::ui::FilePropertyDialogTabActionsView

**class sh::ui::FilePropertyDialogTabActionsView**

A tab view which shows a main widget in the main part and some buttons below.

Subclassed by `sh::ui::qt::QtFilePropertyDialogTabActionsView`, `sh::ui::web::WebFilePropertyDialog::WebFilePropertyDialogTabActionsView`

## Public Functions

`FilePropertyDialogTabActionsView ()`  
 Is intended to be directly constructed from everywhere.

`FilePropertyDialogTabViewContent *content ()`  
 The main widget.

`void setContent (FilePropertyDialogTabViewContent *cnt)`  
 Sets the main widget. .

`QList<QString> buttons ()`  
 The list of buttons.

`void setButtons (QList<QString> buttons)`  
 Sets the list of buttons. .

`void setVisible (bool v) = 0`  
 Sets the view visible or hidden. .

`void onButtonTriggered (std::function<void> int i  
 > fct, QObject *owner = 0)` Sets a handler for a click on one of the buttons (optionally bound to an owner lifetime).

### 10.1.355 Class `sh::ui::FilePropertyDialogTabFactory`

**class** `sh::ui::FilePropertyDialogTabFactory`

Abstract factory for creating property dialog tabs for the selected nodes.

Register an instance of this class with `FilePropertyDialog::addTabFactory` in order to make a property tab implementation available. See also `FilePropertyDialogTabFactoryByFunction`.

Subclassed by `sh::scripting::api::ApiFilePropertyDialogTabFactory`, `sh::ui::FilePropertyDialogTabFactoryByFunction`

#### Public Functions

`std::shared_ptr<sh::ui::FilePropertyDialogTab> construct (std::shared_ptr<sh::ui::FilePropertyDialog> dialog) = 0`

Constructs a fresh instance of a `FilePropertyDialogTab` implementation. .

`~FilePropertyDialogTabFactory ()`

#### Public Static Attributes

`const int REGISTER_FILEPROPERTYDIALOGTAB_INDEX_CORE = 10000`

Base value for display indexes of core (i.e. maximum important) tabs.

Used in `FilePropertyDialog::addTabFactory`.

`const int REGISTER_FILEPROPERTYDIALOGTAB_INDEX_VERYIMPORTANT = 20000`

Base value for display indexes of very important tabs.

Used in `FilePropertyDialog::addTabFactory`.

`const int REGISTER_FILEPROPERTYDIALOGTAB_INDEX_IMPORTANT = 30000`

Base value for display indexes of important tabs.

Used in `FilePropertyDialog::addTabFactory`.

`const int REGISTER_FILEPROPERTYDIALOGTAB_INDEX_NORMAL = 40000`

Base value for display indexes of tabs with medium importance.

Used in `FilePropertyDialog::addTabFactory`.

`const int REGISTER_FILEPROPERTYDIALOGTAB_INDEX_EXOTIC = 50000`

Base value for display indexes of exotic (i.e. less important) tabs.

Used in `FilePropertyDialog::addTabFactory`.

### 10.1.356 Class `sh::ui::FilePropertyDialogTabFactoryByFunction`

**class** `sh::ui::FilePropertyDialogTabFactoryByFunction : public sh::ui::FilePropertyDialogTabFactory`

Implementation of `FilePropertyDialogTabFactory` for making custom `FilePropertyDialogTab` implementations available.

This implementation relies on an external factory function which must be provided to the constructor.

Used internally inside the `REGISTER_FILEPROPERTYDIALOGTAB` macro.

## Public Functions

**FilePropertyDialogTabFactoryByFunction** (std::function<std::shared\_ptr<sh::ui::FilePropertyDialogTab>> fct)

std::shared\_ptr<sh::ui::FilePropertyDialogTab> **construct** (std::shared\_ptr<sh::ui::FilePropertyDialog> dialog)

Constructs a fresh instance of a *FilePropertyDialogTab* implementation. .

## Public Static Attributes

**const int REGISTER\_FILEPROPERTYDIALOGTAB\_INDEX\_CORE** = 10000

Base value for display indexes of core (i.e. maximum important) tabs.

Used in *FilePropertyDialog::addTabFactory*.

**const int REGISTER\_FILEPROPERTYDIALOGTAB\_INDEX\_VERYIMPORTANT** = 20000

Base value for display indexes of very important tabs.

Used in *FilePropertyDialog::addTabFactory*.

**const int REGISTER\_FILEPROPERTYDIALOGTAB\_INDEX\_IMPORTANT** = 30000

Base value for display indexes of important tabs.

Used in *FilePropertyDialog::addTabFactory*.

**const int REGISTER\_FILEPROPERTYDIALOGTAB\_INDEX\_NORMAL** = 40000

Base value for display indexes of tabs with medium importance.

Used in *FilePropertyDialog::addTabFactory*.

**const int REGISTER\_FILEPROPERTYDIALOGTAB\_INDEX\_EXOTIC** = 50000

Base value for display indexes of exotic (i.e. less important) tabs.

Used in *FilePropertyDialog::addTabFactory*.

## Private Members

std::function< std::shared\_ptr< sh::ui::FilePropertyDialogTab >> fct

### 10.1.357 Class sh::ui::FilePropertyDialogTabIconTextBannerView

**class** sh::ui::FilePropertyDialogTabIconTextBannerView : public sh::ui::FilePropertyDialogTabViewContent

A tab view for showing texts and icons combined.

Subclassed by sh::ui::qt::QtFilePropertyDialogTabIconTextBannerView, sh::ui::web::WebFilePropertyDialog::WebFileProperty

## Public Functions

**FilePropertyDialogTabIconTextBannerView** ()

Is intended to be directly constructed from everywhere.

void **clear** () = 0

Clears the contents.

void **insertIcon** (QIcon *icon*, int *sizePt* = 24) = 0

Adds an icon.

void **insertText** (QString *text*) = 0

Adds a text.

### 10.1.358 Class `sh::ui::FilePropertyDialogTabTableView`

**class** `sh::ui::FilePropertyDialogTabTableView` : **public** `sh::ui::FilePropertyDialogTabViewContent`

A tab view for showing key/value pairs.

Subclassed by `sh::ui::qt::QtFilePropertyDialogTabTableView`, `sh::ui::web::WebFilePropertyDialog::WebFilePropertyDialogTabT`

## Public Types

**typedef** QPair<int, int> **ItemIndex**

## Public Functions

**FilePropertyDialogTabTableView** ()

Is intended to be directly constructed from everywhere.

void **setContent** (QList<QPair<QString, QString>> *content*) = 0

Sets the content.

QList<*ItemIndex*> **getSelectedItems** () = 0

Returns the selected cells.

QVariant **getItemValue** (int *r*, int *c*) = 0

Returns a value of a cell.

### 10.1.359 Class `sh::ui::FilePropertyDialogTabTextView`

**class** `sh::ui::FilePropertyDialogTabTextView` : **public** `sh::ui::FilePropertyDialogTabViewContent`

A tab view for showing a text.

Subclassed by `sh::ui::qt::QtFilePropertyDialogTabTextView`, `sh::ui::web::WebFilePropertyDialog::WebFilePropertyDialogTabT`



## Public Functions

**FilePropertyDialogTabTextView()**

Is intended to be directly constructed from everywhere.

void **setContent** (QString *content*) = 0

Sets the content.

### 10.1.360 Class sh::ui::FilePropertyDialogTabViewContent

**class sh::ui::FilePropertyDialogTabViewContent**

Abstract subclass for a tab view content.

Subclassed by *sh::ui::FilePropertyDialogTabIconTextBannerView*, *sh::ui::FilePropertyDialogTabTableView*, *sh::ui::FilePropertyDialogTabTextView*

## Public Functions

**FilePropertyDialogTabViewContent()**

**~FilePropertyDialogTabViewContent()**

### 10.1.361 Class sh::ui::FileView

**class sh::ui::FileView : public QObject**

Abstract base class for a file view implementation (i.e. which shows the content of a directory somehow).

Subclassed by *sh::ui::qt::QtFileViewControl*, *sh::ui::web::WebFileView*

## Public Functions

**FileView()**

**~FileView()**

*sh::ui::ColumnDimensions* \***columnDimensions** () = 0

Returns the column dimensions handler. .

*sh::tools::HistoryTracker*<std::shared\_ptr<const *sh::filesystem::Eurl*>> \***historyTracker** () = 0

Returns the history tracker. .

*FileViewMode* **viewmode** () = 0

Returns the view mode (icons, list, ... ?). .

void **setViewmode** (*FileViewMode* *m*) = 0

Sets the view mode. .

int **index** () = 0

Returns the position of this file view within the panel. .

std::shared\_ptr<*sh::filesystem::FilesystemNode*> **node** ()

Returns the current directory.

void **gotoDir** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> *node*)

Jumps to a new current directory.

Implementations have to call the base class implementation inside.

*sh::filesystem::SizeFormatting* **getSizeFormattingMode** () = 0  
Returns the mode how file sizes are formatted for displaying. .

void **setSizeFormattingMode** (*sh::filesystem::SizeFormatting mode*) = 0  
Sets the mode how file sizes are formatted for displaying. .

bool **hiddenFilesVisible** () = 0  
Returns if hidden files are visible. .

void **setHiddenFilesVisible** (bool *v*) = 0  
Sets the visibility of hidden files. .

int **iconDimension** () = 0  
Returns the icon size (in pixels). .

void **setIconDimension** (int *v*) = 0  
Sets the icon size (in pixels). .

void **setSort** (int *column*, Qt::SortOrder *order*) = 0  
Sets how to sort this view. .

int **sortColumn** () = 0  
Returns the index of the current sort column. .

Qt::SortOrder **sortOrder** () = 0  
Returns the current sort order. .

QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> **selectedNodes** () = 0  
Returns the nodes which the user has selected in this fileview. .

void **setSelection** (const QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> *nodes*) = 0  
Sets the node selection of this fileview. .

QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> **getAllVisibleNodes** () = 0  
Returns a list of all nodes currently listed in this file view. .

void **reload** (bool *skipModel* = false)  
Reloads the data inside this file view.

*sh::filesystem::FilesystemModelFileviewProxy* \***filemodel** ()  
Returns the filesystem model for this view. .

void **setFilemodel** (*sh::filesystem::FilesystemModelFileviewProxy \*model*)  
Sets the filesystem model for this view. .

## Signals

void **selectionChanged** ()  
Triggered when the selection have changed.

void **viewOptionChanged** ()  
Triggered when some view options have changed.

## Private Members

```
sh::filesystem::FilesystemModelFileviewProxy *_model = nullptr
std::shared_ptr<QObject> _visibleviewslifetimecanary = nullptr
std::shared_ptr<sh::filesystem::FilesystemNode> _node = nullptr
```

### 10.1.362 Class sh::ui::LogViewDialog

**class** *sh::ui::LogViewDialog* : **public** *sh::ui::Dialog*

The ‘Log’ dialog..

Subclassed by *sh::ui::qt::QtLogViewDialog*, *sh::ui::web::WebLogViewDialog*

## Public Functions

**LogViewDialog** (QString *headtext*, QString *subheadtxt*, *sh::base::LogSeverity* *minseverity*)

### Parameters

- *headtext*: The header text.
- *subheadtxt*: The 2nd header text.
- *minseverity*: The minimum message severity this dialog shows initially (the user can typically change that later on).

QString **headtext** ()

Returns the header text.

QString **subheadtxt** ()

Returns the 2nd header text.

*sh::base::LogSeverity* **minimumSeverity** ()

Returns the minimum severity.

qint64 **dialogId** ()

Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

bool **isInited** ()

Returns if this dialog is initialized.

Must be called in main thread.

bool **wasAccepted** ()

Returns if this dialog was ‘accepted’ by the user (typically by clicking some ‘OK’ button).

Must be called in main thread.

void **waitClosed** ()

Wait until the user closed the dialog in some way.

May be called in any thread.

void **close** ()

Closes the dialog.

Must be called in main thread.

bool **wasClosed** ()  
Returns if this dialog was closed.  
Must be called in main thread.

*DialogManager* \***manager** ()  
Returns the *DialogManager* which hosts this dialog.

### Private Members

QString **\_headtext**  
QString **\_subheadtxt**  
*sh::base::LogSeverity* **\_minseverity**

## 10.1.363 Class *sh::ui::MainWindow*

**class** *sh::ui::MainWindow*

The abstract main window class. There is one instance of it, and it is also used for other ui parts (e.g. creating some dialogs, ...).

Subclassed by *sh::ui::qt::QtMainWindow*, *sh::ui::web::WebMainWindow*

### Public Functions

**MainWindow** ()

**~MainWindow** ()

void **\_initialize** (*sh::base::SingletonInitializer* \**singletonInitializer*)  
Initializes the main window. For custom initialization logic, see *MainWindow.initialize*. .

int **fileViewCount** () = 0  
Returns the current number of file views. .

void **addFileView** (bool *reinitialize* = false) = 0  
Adds a new file view. .

void **removeFileView** (int *i*) = 0  
Removes a file view. .

*sh::ui::FileView* \***currentFileView** () = 0  
Returns the file view which is currently active (i.e. focussed). .

*sh::ui::FileView* \***getFileView** (int *i*) = 0  
Returns a file view by position index. .

void **fileViewsReloadAll** ()  
Reload all content in each file view.

void **onFileViewOptionsChanged** (std::function<void> int  
> *fcn*, QObject \**owner* = 0) Sets a handler for some view options in a file view changed (optionally bound to an owner lifetime).

void **onCurrentFileViewChanged** (std::function<void>  
> *fcn* QObject \**owner* = 0) Sets a handler for the currently active file view changed (optionally bound to an owner lifetime).

```

void onFileViewCountChanged (std::function<void>
    > fcnQObject *owner = 0)Sets a handler for the number of file views changed (optionally bound to an owner
    lifetime).

void onCurrentDirectoryChanged (std::function<void>
    > fcnQObject *owner = 0)Sets a handler for the current directory in the active file view changed (optionally
    bound to an owner lifetime).

void onGlobalViewOptionsChanged (std::function<void>
    > fcnQObject *owner = 0)Sets a handler for some global view options changed (optionally bound to an
    owner lifetime).

void onCurrentProfileChanged (std::function<void>
    > fcnQObject *owner = 0)Sets a handler for the current profile changed (optionally bound to an owner
    lifetime).

void jumpToEurl (std::shared_ptr<const sh::filesystem::Eurl> eurl)
    Let the current view jump to another location. .

void jumpToNode (std::shared_ptr<sh::filesystem::FilesystemNode> node) = 0
    Let the current view jump to another location. .

void setTitlePattern (QString value)
    Sets the window title pattern. .

QString titlePattern ()
    Returns the window title pattern.

void setCurrentProfile (QString profile)
    Sets the current profile. .

QString currentProfile ()
    Returns the current profile.

void reloadProfile ()
    Reloads the profile data.

void setTreeVisible (bool v)
    Sets the visibility of the directory tree. .

bool treeVisible ()
    Returns the visibility of the directory tree.

void setTreeSticky (bool v)
    Sets if the directory tree should follow the active file view. .

bool treeSticky ()
    Returns if the directory tree follows the active file view.

void setFileDetailsPanelVisible (bool v)
    Sets the visibility of the file details panel. .

bool fileDetailsPanelVisible ()
    Returns the visibility of the file details panel.

std::shared_ptr<sh::filesystem::FilesystemNode> currentDirectory () = 0
    Gets the sh::filesystem::FilesystemNode selected in the current view. .

std::shared_ptr<sh::ui::ActionExecutionInfoPanel> addInfoPanel (int position,
    sh::actions::ActionExecutionInfo
    *info, QColor color = QColor())
    = 0
    Creates a new action execution panel.

```

Let this smart pointer die for removing it.

```
std::shared_ptr<sh::ui::ActionExecutionInfoPanel> addInfoPanel (sh::actions::ActionExecutionInfo
                                                                *info, QColor color = QColor())
```

Creates a new action execution panel.

Let this smart pointer die for removing it.

```
void jumpbarSetTextMode () = 0
    Sets the jumpbar to text input mode. .
```

```
void jumpbarSetButtonMode () = 0
    Sets the jumpbar to button mode. .
```

```
bool jumpbarIsTextMode () = 0
    Returns if the jumpbar is in text input mode. .
```

```
QString toolbarPosition () = 0
    Returns the current toolbar position. .
```

```
void setToolbarPosition (QString pos) = 0
    Sets the toolbar position. .
```

```
QString detailPanelPosition () = 0
    Returns the current detail panel position. .
```

```
void setDetailPanelPosition (QString pos) = 0
    Sets the detail panel position. .
```

```
std::shared_ptr<AboutDialog> showAboutDialog () = 0
    Shows and returns an 'About' dialog. .
```

```
std::shared_ptr<ManageProfilesDialog> showManageProfilesDialog () = 0
    Shows and returns a 'Manage Saved Settings' dialog. .
```

```
std::shared_ptr<OpenWithDialog> showOpenWithDialog (std::shared_ptr<sh::filesystem::FilesystemNode>
                                                    node, QList<std::shared_ptr<sh::tools::filetypes::OpenMethod>>
                                                    openMethods) = 0

    Shows and returns a 'Open With' dialog. .
```

```
std::shared_ptr<StoreProfileDialog> showStoreProfileDialog (std::shared_ptr<const
                                                            sh::filesystem::Eurl> eurl) =
0

    Shows and returns a 'Save Settings' dialog. .
```

```
std::shared_ptr<TuningDialog> showTuningDialog () = 0
    Shows and returns a 'Tuning' dialog. .
```

```
std::shared_ptr<LogViewDialog> showLogViewDialog (QString headtext = QString(),
                                                    QString subheadtxt = QString(),
                                                    sh::base::LogSeverity minseverity =
                                                    sh::base::LogSeverity::_DEFAULT) =
0

    Shows and returns a 'Log' dialog. .
```

```
std::shared_ptr<ManageBookmarksDialog> showManageBookmarksDialog () = 0
    Shows and returns a 'Manage Bookmarks' dialog. .
```

```
std::shared_ptr<FilePropertyDialog> showFilePropertyDialog (QList<std::shared_ptr<sh::filesystem::FilesystemNode>>
                                                            nodes) = 0

    Shows and returns a 'File Properties' dialog. .
```

`std::shared_ptr<ActionExecutionInfoDialog> createActionExecutionInfoDialog (sh::actions::ActionExecutionInfo *info) = 0`

Creates an action execution dialog. .

`std::shared_ptr<ActionExecutionInfoPanel> showErrorPanel ()`

Shows an error panel.

`std::shared_ptr<DialogManager> dialogManager () = 0`

Returns the *DialogManager* which creates and drives *Dialogs* for this main window.

You typically should not need to use it directly.

`void _closeDirectly ()`

Directly begin application shutdown without checks.

`bool closeApp ()`

Check if the application may shut down now, and if so, initiate it.

`bool isCloseable (QString *msg = nullptr)`

Check if the application may shut down now.

`void handleCloseAppRejected (QString rejectmsg) = 0`

React on a rejected application close request (with some feedback message). .

`void handleClose ()`

Do some cleanup steps just when closing starts. Implementations must call base implementation!

`bool isClosing ()`

Returns if the application is currently closing.

## Public Static Functions

*MainWindow* \*mainWindow ()

Returns the instance of this singleton.

`void setMainWindow (MainWindow *mainWindow)`

Sets the main window instance. .

`bool isReady ()`

Returns if this process ui is ready (i.e. is created or runs headless).

`bool runsHeadless ()`

Returns if this process runs headless, i.e. without any ui, just for a background process.

`void _closeDirectly (sh::base::SingletonInitializer *singletonInitializer)`

`QString uiMode ()`

Returns the UI mode (e.g. qt, web).

## Private Functions

`void _refreshIsCloseable ()`

### Private Members

```
bool _closing = false
bool _treeSticky = true
bool _treeVisible = true
bool _detailsPanelVisible = true
QString _titlePattern
QList<std::shared_ptr<sh::ui::ActionExecutionInfoPanel>> _errorpanels
QString _currentProfile
QString _isCloseable
bool _thumbnailrequestongoing = false
bool _thumbnailrequestfollowup = false
int _thumbnailrequestfollowup_width = 0
int _thumbnailrequestfollowup_height = 0
std::function<void ()> _thumbnailrequestfollowup_onBeforeRequest = nullptr
std::function<void (QIcon)> _thumbnailrequestfollowup_onArrived = nullptr
int _thumbnaillastreqwidth = 0
int _thumbnaillastreqheight = 0
quint64 _thumbnaillastrequestid = 0
```

### Private Static Attributes

```
MainWindow *_mainWindow = nullptr
bool _mainWindow_runheadless = false
KillHelperThread *_killthread = nullptr
```

### Friends

```
friend class sh::exceptions::Exception
class KillHelperThread : public QThread
    Helps application shutdown in some situations :).
```

### Public Functions

```
void run ()
```



### 10.1.364 Class `sh::ui::MainWindow::KillHelperThread`

**class** `sh::ui::MainWindow::KillHelperThread` : **public** `QThread`  
 Helps application shutdown in some situations :).

#### Public Functions

void **run** ()

### 10.1.365 Class `sh::ui::ManageBookmarksDialog`

**class** `sh::ui::ManageBookmarksDialog` : **public** `sh::ui::Dialog`  
 The ‘Manage Bookmarks’ dialog.  
 Subclassed by `sh::ui::qt::QtManageBookmarksDialog`, `sh::ui::web::WebManageBookmarksDialog`

#### Public Functions

**ManageBookmarksDialog** ()

qint64 **dialogId** ()  
 Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

bool **isInitiated** ()  
 Returns if this dialog is initialized.

Must be called in main thread.

bool **wasAccepted** ()  
 Returns if this dialog was ‘accepted’ by the user (typically by clicking some ‘OK’ button).

Must be called in main thread.

void **waitClosed** ()  
 Wait until the user closed the dialog in some way.

May be called in any thread.

void **close** ()  
 Closes the dialog.

Must be called in main thread.

bool **wasClosed** ()  
 Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \***manager** ()  
 Returns the *DialogManager* which hosts this dialog.

### 10.1.366 Class `sh::ui::ManageProfilesDialog`

**class** `sh::ui::ManageProfilesDialog` : **public** `sh::ui::Dialog`

The ‘Manage Saved Settings’ dialog.

Subclassed by `sh::ui::qt::QtManageProfilesDialog`, `sh::ui::web::WebManageProfilesDialog`

#### Public Functions

**ManageProfilesDialog** ()

qint64 **dialogId** ()

Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

bool **isInited** ()

Returns if this dialog is initialized.

Must be called in main thread.

bool **wasAccepted** ()

Returns if this dialog was ‘accepted’ by the user (typically by clicking some ‘OK’ button).

Must be called in main thread.

void **waitClosed** ()

Wait until the user closed the dialog in some way.

May be called in any thread.

void **close** ()

Closes the dialog.

Must be called in main thread.

bool **wasClosed** ()

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \***manager** ()

Returns the *DialogManager* which hosts this dialog.

### 10.1.367 Class `sh::ui::OpenWithDialog`

**class** `sh::ui::OpenWithDialog` : **public** `sh::ui::Dialog`

The ‘Open With’ dialog.

Subclassed by `sh::ui::qt::QtOpenWithDialog`, `sh::ui::web::WebOpenWithDialog`

## Public Functions

**OpenWithDialog** (std::shared\_ptr<sh::filesystem::FilesystemNode> *node*,  
QList<std::shared\_ptr<sh::tools::filetypes::OpenMethod>> *openMethods*)

### Parameters

- *node*: The file which is later to be opened.
- *openMethods*: The open-methods (programs for opening the file) to present for choice.

std::shared\_ptr<sh::tools::filetypes::OpenMethod> **chosenMethod** () = 0  
Returns the chosen open-method (program for opening the file).

bool **rememberForMimetype** () = 0  
Returns if the chosen method is checked to be remembered for the same mime-type in the future.

std::shared\_ptr<const sh::filesystem::Eurl> **rememberForDirectory** () = 0  
If it's checked for the chosen method to be remembered for some subdirectory in the future, it returns the Eurl of this subdirectory, otherwise nullptr.

bool **rememberForFile** () = 0  
Returns if the chosen method is checked to be remembered for the same file in the future.

std::shared\_ptr<sh::filesystem::FilesystemNode> **node** ()  
Returns the file node which is later to be opened.

QList<std::shared\_ptr<tools::filetypes::OpenMethod>> **openMethods** ()  
Returns the open-methods (programs for opening the file) to present for choice.

qint64 **dialogId** ()  
Returns the dialog id.  
  
Each instance has an id unique in the complete Shallot process lifetime.  
  
Must be called in main thread.

bool **isInitied** ()  
Returns if this dialog is initialized.  
  
Must be called in main thread.

bool **wasAccepted** ()  
Returns if this dialog was 'accepted' by the user (typically by clicking some 'OK' button).  
  
Must be called in main thread.

void **waitClosed** ()  
Wait until the user closed the dialog in some way.  
  
May be called in any thread.

void **close** ()  
Closes the dialog.  
  
Must be called in main thread.

bool **wasClosed** ()  
Returns if this dialog was closed.  
  
Must be called in main thread.

DialogManager \***manager** ()  
Returns the *DialogManager* which hosts this dialog.

### Private Members

`std::shared_ptr<sh::filesystem::FilesystemNode> _node`  
`QList<std::shared_ptr<sh::tools::filetypes::OpenMethod>> _openMethods`

## 10.1.368 Class `sh::ui::SearchPanelAbstractEditor`

**class** `sh::ui::SearchPanelAbstractEditor`

Abstract base class for editor widgets in a search panel.

Subclassed by `sh::ui::SearchPanelDateTimeEditor`, `sh::ui::SearchPanelLabelEditor`,  
`sh::ui::SearchPanelSpacerEditor`, `sh::ui::SearchPanelTextEditor`

### Public Functions

`SearchPanelAbstractEditor()` = default

`~SearchPanelAbstractEditor()` = default

bool `isEnabled()` = 0  
Returns if the widget is enabled. .

void `setEnabled(bool v)` = 0  
Sets if the widget is enabled. .

## 10.1.369 Class `sh::ui::SearchPanelButton`

**class** `sh::ui::SearchPanelButton`

A button in the button bar of a search panel.

Subclassed by `sh::ui::qt::QtSearchPanelButton`

### Public Functions

`SearchPanelButton()`

`~SearchPanelButton()`

void `setButtonText(QString txt)` = 0  
Sets the button text. .

void `setMenuSelection(int i)` = 0  
Sets the currently selected menu item (for buttons with menus). .

### 10.1.370 Class `sh::ui::SearchPanelConfiguration`

**class** `sh::ui::SearchPanelConfiguration`

Configuration for a search panel population.

It is populated with ui controls by the chosen search criterion. Later on those controls are used for updating the configuration data.

Subclassed by `sh::ui::qt::QtSearchPanelConfiguration`

#### Public Functions

`SearchPanelConfiguration()`

`~SearchPanelConfiguration()`

`SearchPanelButton *addMenuButton (QString text, QStringList menu, std::function<void> int  
> onChanged = 0` Adds and returns a button for a menu. .

`SearchPanelButton *addActionButton (QString text, std::function<void>  
> action = 0` Adds and returns a button for an action. .

`SearchPanelTextEditor *addTextEditor () = 0`  
Adds and returns a text editor. .

`SearchPanelDateTimeEditor *addDateTimeEditor () = 0`  
Adds and returns a date/time editor. .

`SearchPanelLabelEditor *addLabel () = 0`  
Adds and returns a label. .

`SearchPanelSpacerEditor *addSpacer () = 0`  
Adds and returns a spacer. .

`SearchPanelAbstractEditor *getEditorAt (int i) = 0`  
Returns the editor widget at a given position. .

void `onDestroyed` (std::function<void>  
> `fcn`QObject \*owner = 0) Sets a handler for panel removal (optionally bound to an owner lifetime).

#### Public Members

`_SearchPanelConfiguration_HelperQObject myqobject`

### 10.1.371 Class `sh::ui::SearchPanelDateTimeEditor`

**class** `sh::ui::SearchPanelDateTimeEditor : public sh::ui::SearchPanelAbstractEditor`

A date/time picker for search panel usage.

Subclassed by `sh::ui::qt::QtSearchPanelAbstractEditor< QDateTimeEdit, SearchPanelDateTimeEditor >`

### Public Functions

**SearchPanelDateTimeEditor** () = default

QDateTime **datetime** () = 0

Returns the selected date/time. .

void **setDatetime** (QDateTime *s*) = 0

Sets the selected date/time.

bool **isWidgetEnabled** () = 0

Returns if the widget is enabled. .

void **setWidgetEnabled** (bool *v*) = 0

Sets if the widget is enabled. .

## 10.1.372 Class sh::ui::SearchPanelLabelEditor

**class** *sh::ui::SearchPanelLabelEditor* : **public** *sh::ui::SearchPanelAbstractEditor*

A text label for search panel usage.

Subclassed by *sh::ui::qt::QtSearchPanelAbstractEditor< QLabel, sh::ui::SearchPanelLabelEditor >*

### Public Functions

**SearchPanelLabelEditor** () = default

QString **textContent** () = 0

Returns the text content. .

void **setTextContent** (QString *s*) = 0

Sets the text content. .

*SearchPanelAbstractEditor* \***focusProxyEditor** () = 0

Returns the focus proxy widget (which focus gets redirected to in some situations). .

void **setFocusProxyEditor** (*SearchPanelAbstractEditor* \**w*) = 0

Sets the focus proxy widget (which focus gets redirected to in some situations). .

bool **isWidgetEnabled** () = 0

Returns if the widget is enabled. .

void **setWidgetEnabled** (bool *v*) = 0

Sets if the widget is enabled. .

## 10.1.373 Class sh::ui::SearchPanelSpacerEditor

**class** *sh::ui::SearchPanelSpacerEditor* : **public** *sh::ui::SearchPanelAbstractEditor*

A spacer for search panel usage.

Subclassed by *sh::ui::qt::QtSearchPanelAbstractEditor< QLabel, sh::ui::SearchPanelSpacerEditor >*

## Public Functions

**SearchPanelSpacerEditor** () = default

bool **isEnabled** () = 0  
Returns if the widget is enabled. .

void **setEnabled** (bool v) = 0  
Sets if the widget is enabled. .

### 10.1.374 Class sh::ui::SearchPanelTextEditor

**class** *sh::ui::SearchPanelTextEditor* : **public** *sh::ui::SearchPanelAbstractEditor*

A single line text editor for search panel usage.

Subclassed by *sh::ui::qt::QtSearchPanelAbstractEditor< QLineEdit, sh::ui::SearchPanelTextEditor >*

## Public Functions

**SearchPanelTextEditor** () = default

QString **textContent** () = 0  
Returns the text content. .

void **settextContent** (QString s) = 0  
Sets the text content. .

QString **placeholderDescription** () = 0  
Returns the placeholder description text (visible if field is empty). .

void **setPlaceholderDescription** (QString s) = 0  
Sets the placeholder description text (visible if field is empty). .

bool **isEnabled** () = 0  
Returns if the widget is enabled. .

void **setEnabled** (bool v) = 0  
Sets if the widget is enabled. .

### 10.1.375 Class sh::ui::StoreProfileDialog

**class** *sh::ui::StoreProfileDialog* : **public** *sh::ui::Dialog*

The ‘Save Settings’ dialog.

Subclassed by *sh::ui::qt::QtStoreProfileDialog*, *sh::ui::web::WebStoreProfileDialog*

## Public Functions

**StoreProfileDialog** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*)

### Parameters

- *eurl*: The current directory this dialog shall work on.

std::shared\_ptr<const *sh::filesystem::Eurl*> **eurl** ()

Returns the current directory this dialog shall work on.

QList<*SettingEntry*> **checkedSettings** () = 0

Returns the list of settings the user has checked to store.

QString **profileName** () = 0

Returns the profile the user has chosen to store settings for.

bool **withSubDirectories** () = 0

Returns if the user has chosen to apply the checked settings also for subdirectories.

QStringList **inheritsFrom** () = 0

Returns the list of profiles the user has chosen to inherit from.

bool **hasGlobal** () = 0

Returns if the user has chosen to apply the checked settings for each directory (instead of the current directory).

qint64 **dialogId** ()

Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

bool **isInited** ()

Returns if this dialog is initialized.

Must be called in main thread.

bool **wasAccepted** ()

Returns if this dialog was 'accepted' by the user (typically by clicking some 'OK' button).

Must be called in main thread.

void **waitClosed** ()

Wait until the user closed the dialog in some way.

May be called in any thread.

void **close** ()

Closes the dialog.

Must be called in main thread.

bool **wasClosed** ()

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \***manager** ()

Returns the *DialogManager* which hosts this dialog.



### Private Members

```
std::shared_ptr<const sh::filesystem::Eurl> _eurl
```

```
class SettingEntry
```

A storable entry in a ‘Save Settings’ dialog.

### Public Functions

```
SettingEntry (sh::settings::Setting *setting, int onlyInFileview = -1)
```

```
sh::settings::Setting *setting ()
```

Returns the setting to store.

```
int fileview ()
```

Returns the index of the file view to store this setting for (or -1 for all).

```
bool isForFileview ()
```

Returns if this setting is to be stored for a particular file view (instead of for all).

### Private Members

```
sh::settings::Setting *_setting
```

```
int _fileview
```

## 10.1.376 Class sh::ui::StoreProfileDialog::SettingEntry

```
class sh::ui::StoreProfileDialog::SettingEntry
```

A storable entry in a ‘Save Settings’ dialog.

### Public Functions

```
SettingEntry (sh::settings::Setting *setting, int onlyInFileview = -1)
```

```
sh::settings::Setting *setting ()
```

Returns the setting to store.

```
int fileview ()
```

Returns the index of the file view to store this setting for (or -1 for all).

```
bool isForFileview ()
```

Returns if this setting is to be stored for a particular file view (instead of for all).

### Private Members

```
sh::settings::Setting *_setting
```

```
int _fileview
```

### 10.1.377 Class `sh::ui::TuningDialog`

**class** `sh::ui::TuningDialog` : **public** `sh::ui::Dialog`

The ‘Tuning’ dialog.

Subclassed by `sh::ui::qt::QtTuningDialog`, `sh::ui::web::WebTuningDialog`

#### Public Functions

**TuningDialog** ()

qint64 **dialogId** ()

Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

bool **isInited** ()

Returns if this dialog is initialized.

Must be called in main thread.

bool **wasAccepted** ()

Returns if this dialog was ‘accepted’ by the user (typically by clicking some ‘OK’ button).

Must be called in main thread.

void **waitClosed** ()

Wait until the user closed the dialog in some way.

May be called in any thread.

void **close** ()

Closes the dialog.

Must be called in main thread.

bool **wasClosed** ()

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \***manager** ()

Returns the *DialogManager* which hosts this dialog.

### 10.1.378 Class `sh::ui::_ActionExecutionInfoPanel_HelperQObject`

**class** `sh::ui::_ActionExecutionInfoPanel_HelperQObject` : **public** `QObject`

A helper for some signal/slot stuff of in *ActionExecutionInfoPanel*.

### Public Functions

```
void _emit_clicked ()  
void _emit_visibilityChanged ()
```

### Signals

```
void clicked ()  
void visibilityChanged ()
```

## 10.1.379 Class `sh::ui::_FilePropertyDialogTabActionsView_HelperQObject`

**class** `sh::ui::_FilePropertyDialogTabActionsView_HelperQObject` : **public** `QObject`  
A helper for some signal/slot stuff of in *FilePropertyDialogTabActionsView*.

### Public Functions

```
void _emit_buttonTriggered (int i)
```

### Signals

```
void buttonTriggered (int i)
```

## 10.1.380 Class `sh::ui::_MainWindow_HelperQObject`

**class** `sh::ui::_MainWindow_HelperQObject` : **public** `QObject`  
A helper for some signal/slot stuff of in *MainWindow*.

### Public Functions

```
void _emit_currentDirectoryChanged ()  
void _emit_globalViewOptionsChanged ()  
void _emit_currentProfileChanged ()  
void _emit_currentFileViewChanged ()  
void _emit_fileViewCountChanged ()  
void _emit_fileViewOptionsChanged (int i)
```

### Signals

```
void currentDirectoryChanged ()  
void globalViewOptionsChanged ()  
void currentProfileChanged ()  
void currentFileViewChanged ()  
void fileViewCountChanged ()  
void fileViewOptionsChanged (int)
```

#### 10.1.381 Class `sh::ui::_SearchPanelConfiguration_HelperQObject`

```
class _SearchPanelConfiguration_HelperQObject : public QObject  
    A helper for some signal/slot stuff of in SearchPanelConfiguration.
```

#### 10.1.382 Class `sh::ui::qt::LineEditWithKeyboardShortcuts`

```
class sh::ui::qt::LineEditWithKeyboardShortcuts : public QLineEdit  
    Helper widget of QtJumpBar.
```

### Public Functions

```
LineEditWithKeyboardShortcuts (QWidget *parent = 0)
```

### Signals

```
void enterPressed ()  
void escapePressed ()
```

#### 10.1.383 Class `sh::ui::qt::QtAboutDialog`

```
class sh::ui::qt::QtAboutDialog : public sh::ui::qt::QtDialog, public sh::ui::AboutDialog  
    Qt based about dialog.
```

### Public Functions

```
QtAboutDialog ()  
~QtAboutDialog ()  
qint64 dialogId ()  
    Returns the dialog id.  
    Each instance has an id unique in the complete Shallot process lifetime.  
    Must be called in main thread.
```

bool **isInited** ()  
Returns if this dialog is initialized.  
Must be called in main thread.

bool **wasAccepted** ()  
Returns if this dialog was ‘accepted’ by the user (typically by clicking some ‘OK’ button).  
Must be called in main thread.

void **waitClosed** ()  
Wait until the user closed the dialog in some way.  
May be called in any thread.

void **close** ()  
Closes the dialog.  
Must be called in main thread.

bool **wasClosed** ()  
Returns if this dialog was closed.  
Must be called in main thread.

*DialogManager* \***manager** ()  
Returns the *DialogManager* which hosts this dialog.

### Private Members

*Ui*::QtAboutDialog \***ui**

### Private Slots

void **on\_btnClose\_clicked** ()  
void **on\_btnLicense\_clicked** ()  
void **on\_btnHomepage\_clicked** ()

## 10.1.384 Class `sh::ui::qt::QtActionExecutionInfoDialog`

**class** `sh::ui::qt::QtActionExecutionInfoDialog` : **public** QDialog, **public** `sh::ui::ActionExecutionInfoDialog`  
Qt progress dialog for action executions.

### Public Types

**enum** **MessageBoxButton**  
Buttons in a message box from *ActionExecutionUserFeedback*.  
*Values:*

**enumerator** **NONE** = 0  
**enumerator** **OK** = 1 << 0  
**enumerator** **Continue** = 1 << 1  
**enumerator** **Cancel** = 1 << 2

```
enumerator Retry = 1 << 3
enumerator Yes = 1 << 4
enumerator No = 1 << 5
```

## Public Functions

```
QtActionExecutionInfoDialog (sh::actions::ActionExecutionInfo *info, QWidget *parent = 0)
~QtActionExecutionInfoDialog ()

void setDetails (QString fv, QString fob, QString tv, QString tob)
    Sets the item details text ('from a/foo.jpg', 'to b/foo.jpg'). .

void setHead (QString txt)
    Sets the header text. .

void setProgress (bool isDeterminate, quint64 done, quint64 all, QString text)
    Sets the progress. .

int messageBox (QString text, QList<QString> answers, QString icon = QString(), int defaultanswer =
    -1, int cancelanswer = -1, QList<QString> answericons = QList<QString>())

int inputBox (QString text, QList<QString> answers, QString *value, QString icon = QString(), int de-
    faultanswer = -1, int cancelanswer = -1, int valuePreselectFrom = -1, int valuePreselectTo
    = -1)

int multilineInputBox (QString text, QList<QString> answers, QString *value, QString icon =
    QString(), int defaultanswer = -1, int cancelanswer = -1)

int simpleChooserGridform (QString text, GridformEntries *entries, QStringList answers, int de-
    faultanswer = -1, int cancelanswer = -1)

bool credentialsDialog (QString text, bool showDomain, bool showUsername, bool showPass-
    word, bool showAnonymous, bool showRemember, QString *domain,
    QString *username, QString *password, bool *isAnonymous, bool *is-
    Remember)

bool unixPermissionsDialog (bool *userMayRead, bool *userMayWrite, bool *userMayExecute,
    bool *groupMayRead, bool *groupMayWrite, bool *groupMayEx-
    ecute, bool *othersMayRead, bool *othersMayWrite, bool *others-
    MayExecute, bool *sticky, bool *setuid, bool *setgid, QStringList
    users, QStringList groups, QString *ownerUser, QString *owner-
    Group)

void setLogicallyVisible (bool v)
    Sets if this dialog is logically visible (i.e. set visible by the action). .

void setBackground (bool v)
    Sets if this dialog is currently in background mode (i.e. not visible). .

void setForceForeground (bool v)
    Sets if this dialog is currently forced to be visible in foreground. .

bool isLogicallyVisible ()
    Returns if this dialog is logically visible (i.e. set visible by the action).

bool isBackground ()
    Returns if this dialog is currently in background mode (i.e. not visible).

QString simpleInputBox (QString text, QString deflt, int valuePreselectFrom = -1, int valuePrese-
    lectTo = -1)
```

```
int simpleMessageBox (QString text, int buttons = (int)MessageBoxButton::OK, QString icon =
    QString(), int defaultbutton = (MessageBoxButton)0, int cancelbutton =
    (MessageBoxButton)0)
```

### Private Functions

```
void _computevisibility ()
```

### Private Members

```
Ui::QtActionExecutionInfoDialog *ui
```

```
sh::ui::qt::feedbackpanels::FeedbackPanel *currentFeedbackPanel = 0
```

### Private Slots

```
void on_btnCancel_clicked ()
```

```
void on_btnBackground_clicked ()
```

## 10.1.385 Class *sh::ui::qt::QtActionExecutionInfoPanel*

```
class sh::ui::qt::QtActionExecutionInfoPanel : public QWidget, public sh::ui::ActionExecutionInfoPanel
    Qt status bar info-panel for an action execution.
```

### Public Functions

```
QtActionExecutionInfoPanel (sh::actions::ActionExecutionInfo *info, QColor color, QWidget
    *parent = 0)
```

```
void setLabel (QString s)
    Sets the label text. .
```

```
void setWidth (int width)
    Sets the panel with (in pixels). .
```

```
QSize sizeHint () const
```

```
void setProgress (bool isProgressDeterminate, quint64 progressDone, quint64 progressAll)
    Sets the progress. .
```

```
void setForceForeground (bool v)
    Sets if the associated action is currently forced to be visible in foreground. .
```

```
void setPanelVisible (bool v)
    Sets if the panel is visible. .
```

```
void onClicked (std::function<void>)
    > fcnQObject *owner = 0 Sets a handler for a click on the button (optionally bound to an owner lifetime).
```

```
void onDestroyed (std::function<void>)
    > fcnQObject *owner = 0 Sets a handler for panel removal (optionally bound to an owner lifetime).
```

```
void onVisibilityChanged (std::function<void>)
    > fcnQObject *owner = 0 Sets a handler for panel visibility changes (optionally bound to an owner lifetime).
```

### Private Functions

```
void _check_indeterminateanimationtimer_enable()
```

### Private Members

```
QString lb11
bool _progressDeterminate = false
int _indeterminateAnimationCounter = 0
int _width
QTimer *_indeterminateAnimationTimer
quint64 _progressAll
quint64 _progressDone
bool _isforeground_forced = false
sh::actions::ActionExecutionInfo *info
QPixmap gradient
QColor colorProgress2
QColor colorProgress1
QColor colorBase
QColor colorFrame
QColor colorText
QColor colorTextDark
```

## 10.1.386 Class sh::ui::qt::QtActionMenu

```
class sh::ui::qt::QtActionMenu : public QMenu
    Qt based action menu used for context menus and in the toolbar.
```

### Public Functions

```
QtActionMenu (QWidget *parent = 0)
void setHeader (QString t)
QList<QAction*> allActions ()
void clearAllActions ()
void removeActionAt (int i)
QAction *addNewAction (int i = -1)
QAction *addNewSubMenu (sh::ui::qt::QtActionMenu **qsubmenu, int i = -1)
QAction *addNewSeparator (int i = -1)
QAction *addNewHeader (int i = -1)
bool actionIsHeader (QAction *a)
```



### Private Functions

```
void _observeAction (QAction *a)
void _correctMenuPosition ()
```

### Private Members

```
QColor brandingcolor
QFont _boldfont
QFont _normalfont
int _origMenuPosX = -1
int _origMenuPosY = -1
QString _header
int _cntInternalActions
```

## 10.1.387 Class sh::ui::qt::QtActionMenuHandler

```
class sh::ui::qt::QtActionMenuHandler
```

A handler which reflects the *sh::actions* objects to a graphical menu (and keeps that up-to-date).

### Public Functions

```
QtActionMenuHandler (std::shared_ptr<sh::actions::ActionInstantiation> ai,
                    std::shared_ptr<QtActionMenu> menu)
```

### Private Functions

```
void _markDefault2 ()
void _append_actions (sh::ui::qt::QtActionMenu *menu, QList<std::shared_ptr<sh::actions::ActionInstantiation>>
                    acts, std::shared_ptr<sh::actions::ActionCategory> category)
```

### Private Members

```
QList<std::shared_ptr<sh::actions::ActionInstantiation>> selacts
QList<std::shared_ptr<sh::actions::ActionInstantiation>> diracts
QHash<QAction*, std::shared_ptr<sh::actions::AbstractActionItem>> qaction2action
std::weak_ptr<QtActionMenu> menu
```

### Private Static Functions

```
std::shared_ptr<sh::actions::ActionActionItem> _getDefaultAction (QList<std::shared_ptr<sh::actions::AbstractActionItem>
                                                                    actionList)

void _markDefault (std::weak_ptr<sh::actions::SubmenuActionItem> itmSubmenu,
                  sh::ui::qt::QtActionMenu *menu)

QAction *_createAndConnectAction (sh::actions::AbstractActionItem *itm,
                                 sh::ui::qt::QtActionMenu *menu, std::function<void>
                                 > onChangedQObject *onchangedbuddy = 0)

void _applyPropertiesToQAction (sh::actions::AbstractActionItem *itm, QAction *_widgetac-
                              tion)

void _updateSubmenu (sh::actions::SubmenuActionItem *itm, QtActionMenu *menu,
                    std::function<void>
                    > onChanged = 0)
```

### Friends

```
friend class QtToolBarButtonHandler
```

## 10.1.388 Class sh::ui::qt::QtDialog

```
class sh::ui::qt::QtDialog : public QDialog
```

Abstract base class for a qt based dialog. Typically used for also implementing some *sh::ui::Dialog*.

Subclassed by *sh::ui::qt::QtAboutDialog*, *sh::ui::qt::QtExceptionDialog*, *sh::ui::qt::QtFilePropertyDialog*, *sh::ui::qt::QtLogViewDialog*, *sh::ui::qt::QtManageBookmarksDialog*, *sh::ui::qt::QtManageProfilesDialog*, *sh::ui::qt::QtOpenWithDialog*, *sh::ui::qt::QtStoreProfileDialog*, *sh::ui::qt::QtTuningDialog*

### Public Functions

```
QtDialog()
```

## 10.1.389 Class sh::ui::qt::QtDialogManager

```
class sh::ui::qt::QtDialogManager : public sh::ui::DialogManager
```

A *DialogManager* used in Qt ui.

### Public Functions

```
std::shared_ptr<Dialog> getDialogById (qint64 id)
```

Returns a *Dialog* by dialog id.

Must be called in main thread.

```
QList<std::shared_ptr<Dialog>> getAllDialogs ()
```

Returns a list of all dialogs currently shown (in order of creation).

Must be called in main thread.

`QList<qint64> getAllDialogIds ()`

Returns a list of dialog ids of all dialogs currently shown (in order of creation).

Must be called in main thread.

`template<class TDlg, typename ...Args>`

`std::shared_ptr<TDlg> createAndShowDialog (Args... args)`

Creates and shows a *Dialog* by class and constructor parameters.

Those dialogs (*TDlg*) have to implement *Dialog* (typically a subclass of it, representing a particular dialog, like *FilePropertyDialog*), and also some ui mode (e.g. qt, web) specific class (depends on that ui mode).

You typically should not have to use it outside of *MainWindow* or subclasses. The *MainWindow* interface allows to create all available dialogs in a ui mode independent way.

May be called in any thread.

**Return** The created *Dialog* instance.

### Private Functions

`void showDialog (std::shared_ptr<Dialog> dialog) override`

This method implements the ui mode specific mechanism for showing a dialog.

Must be called in main thread.

## 10.1.390 Class `sh::ui::qt::QtExceptionDialog`

`class sh::ui::qt::QtExceptionDialog : public sh::ui::qt::QtDialog, public sh::ui::ExceptionDialog`  
Qt based exception dialog.

### Public Functions

`QtExceptionDialog (QString error1, QString error2, QString details, QString icon, bool mayRetry,  
bool mayClose, bool mayCancel, bool showLoglabelAndDetails)`

`~QtExceptionDialog ()`

`ExceptionDialogResult answer () override`

Returns the answer the user has given in this dialog.

`QString error1 ()`

Returns the 1st error text.

`QString error2 ()`

Returns the 2nd error text.

`QString details ()`

Returns the error details.

`QString icon ()`

Returns the icon (as name resolveable by *sh::base::IconManager*).

`bool mayRetry ()`

Returns if it's allowed to retry.

`bool mayClose ()`

Returns if it's allowed to just close and continue without closing Shallot.

bool **mayCancel** ()  
Returns if it's allowed to cancel, i.e. throwing a *sh::exceptions::CancelException*.

bool **showLoglabelAndDetails** ()  
Returns if the dialog shall allow to read the details.

qint64 **dialogId** ()  
Returns the dialog id.  
  
Each instance has an id unique in the complete Shallot process lifetime.  
  
Must be called in main thread.

bool **isInited** ()  
Returns if this dialog is initialized.  
  
Must be called in main thread.

bool **wasAccepted** ()  
Returns if this dialog was 'accepted' by the user (typically by clicking some 'OK' button).  
  
Must be called in main thread.

void **waitClosed** ()  
Wait until the user closed the dialog in some way.  
  
May be called in any thread.

void **close** ()  
Closes the dialog.  
  
Must be called in main thread.

bool **wasClosed** ()  
Returns if this dialog was closed.  
  
Must be called in main thread.

*DialogManager* \***manager** ()  
Returns the *DialogManager* which hosts this dialog.

## Private Members

*Ui::QtExceptionDialog* \***ui**

*ExceptionDialogResult* **\_answer** = *ExceptionDialogResult::Shutdown*

## Private Slots

void **on\_btnExit\_clicked** ()  
void **on\_btnClose\_clicked** ()  
void **on\_btnCancel\_clicked** ()  
void **on\_btnRetry\_clicked** ()  
void **on\_btnShowLog\_clicked** ()  
void **on\_btnExitAndSaveLog\_clicked** ()  
void **on\_btnShowDetails\_toggled** (bool *checked*)

### 10.1.391 Class `sh::ui::qt::QtFileDetailsPanel`

**class** `sh::ui::qt::QtFileDetailsPanel` : **public** `QWidget`

The details panel.

Can be shown in the main window. It presents detail information about the selected file.

#### Public Functions

`QtFileDetailsPanel` (`QWidget *parent = 0`)

void **setNodes** (`QList<std::shared_ptr<sh::filesystem::FilesystemNode>> nodes`)

void **setOrientation** (`Qt::Orientation orientation`)

`QSize` **sizeHint** () **const override**

**~QtFileDetailsPanel** ()

#### Private Members

int **PADDINGX**

int **PADDINGY**

`QList<std::shared_ptr<sh::filesystem::FilesystemNode>> _nodes`

`QList<std::shared_ptr<sh::paneldetails::PanelDetail>> _panelDetails`

`QList<DetailPlacement> _placements`

`QFont` **fontNormal**

`QFont` **fontBold**

`QPixmap` **\_widgetimagecache**

`QTimer` **\_widgetimagecachetimer**

`Qt::Orientation` **\_orientation** = `Qt::Horizontal`

**class** `DetailPlacement`

#### Public Functions

**DetailPlacement** (`int x`, `int y`, `int w`, `int h`, `QList<DetailRowPlacement> rowplacements`,  
`std::shared_ptr<sh::paneldetails::PanelDetail> detail`)

#### Public Members

`std::shared_ptr<sh::paneldetails::PanelDetail>` **detail**

int **x**

int **y**

int **w**

int **h**

`QList<DetailRowPlacement>` **rowplacements**

```
class DetailRowPlacement
```

### Public Functions

```
DetailRowPlacement (int h, int contentx, int contenty, QList<int> elementwidths)
```

```
DetailRowPlacement () = default
```

```
DetailRowPlacement (const DetailRowPlacement&) = default
```

### Public Members

```
int h
```

```
int contentx
```

```
int contenty
```

```
QList<int> elementwidths
```

## 10.1.392 Class sh::ui::qt::QtFileDetailsPanel::DetailPlacement

```
class sh::ui::qt::QtFileDetailsPanel::DetailPlacement
```

### Public Functions

```
DetailPlacement (int x, int y, int w, int h, QList<DetailRowPlacement> rowplacements,  
std::shared_ptr<sh::paneldetails::PanelDetail> detail)
```

### Public Members

```
std::shared_ptr<sh::paneldetails::PanelDetail> detail
```

```
int x
```

```
int y
```

```
int w
```

```
int h
```

```
QList<DetailRowPlacement> rowplacements
```

## 10.1.393 Class sh::ui::qt::QtFileDetailsPanel::DetailRowPlacement

```
class sh::ui::qt::QtFileDetailsPanel::DetailRowPlacement
```

## Public Functions

**DetailRowPlacement** (int *h*, int *contentx*, int *contenty*, QList<int> *elementwidths*)

**DetailRowPlacement** () = default

**DetailRowPlacement** (const *DetailRowPlacement*&) = default

## Public Members

int **h**

int **contentx**

int **contenty**

QList<int> **elementwidths**

### 10.1.394 Class sh::ui::qt::QtFileIconview

**class** *sh::ui::qt::QtFileIconview*: public QListView, public *sh::ui::qt::QtFileView*  
Icon view for the contents of one directory.

## Public Functions

**QtFileIconview** (QWidget *\*parent* = 0)

Constructed only by the infrastructure and made available otherwise.

void **setThumbDimension** (double *size*)

Sets the size of the thumbnail image.

void **setBackgroundColor** (QString *c*)

QString **backgroundColor** ()

QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> **selectedNodes** ()

void **setSelection** (const QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> *nodes*)

std::shared\_ptr<*sh::filesystem::FilesystemNode*> **node** ()

QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> **getAllVisibleNodes** ()

void **setSizeFormatting** (*sh::filesystem::SizeFormatting* *v*)

void **gotoDir** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> *n*)

void **setHiddenFilesVisible** (bool *value*)

QThread **\*thread** ()

void **configure** (*sh::ui::qt::QtFileViewControl* *\*viewctrl*)

void **focus** ()

void **setSort** (int *column*, Qt::SortOrder *order*)

QObject **\*as\_qobject** ()

QWidget **\*as\_qwidget** ()

QAbstractItemView **\*as\_qabstractitemview** ()

```
sh::ui::qt::QtFileViewControl *control ()
```

### Private Members

```
int _deleg_w = 0  
int _deleg_h = 0  
int _dsx = 0  
int _dsy = 0  
int _lineheight = 0
```

## 10.1.395 Class `sh::ui::qt::QtFileIconview::MyItemDelegate`

```
class sh::ui::qt::QtFileIconview::MyItemDelegate : public QStyledItemDelegate
```

### Public Functions

```
MyItemDelegate (QObject *parent)  
QSize sizeHint (const QStyleOptionViewItem &option, const QModelIndex &index) const  
    override  
void paint (QPainter *painter, const QStyleOptionViewItem &option, const QModelIndex &index) const override
```

### Private Functions

```
int _getVisibleCharCount (QString text, int width, QFontMetrics *fm, bool forward) const
```

### Private Members

```
const QString ELLIPSIS = "..."
```

## 10.1.396 Class `sh::ui::qt::QtFileList`

```
class sh::ui::qt::QtFileList : public QTreeView, public sh::ui::qt::QtFileView  
    List view for the contents of one directory.
```

### Public Functions

```
QtFileList (QWidget *parent = 0)  
    Constructed only by the infrastructure and made available otherwise.  
void setSort (int column, Qt::SortOrder order) override  
void setBackgroundColor (QString c)  
QString backgroundColor ()  
QList<std::shared_ptr<sh::filesystem::FilesystemNode>> selectedNodes ()  
void setSelection (const QList<std::shared_ptr<sh::filesystem::FilesystemNode>> nodes)
```



```

std::shared_ptr<sh::filesystem::FilesystemNode> node ()
QList<std::shared_ptr<sh::filesystem::FilesystemNode>> getAllVisibleNodes ()
void setSizeFormatting (sh::filesystem::SizeFormatting v)
void gotoDir (std::shared_ptr<sh::filesystem::FilesystemNode> n)
void setHiddenFilesVisible (bool value)
QThread *thread ()
void configure (sh::ui::qt::QtFileViewControl *viewctrl)
void focus ()
QObject *as_qobject ()
QWidget *as_qwidget ()
QAbstractItemView *as_qabstractitemview ()
sh::ui::qt::QtFileViewControl *control ()

```

### Private Functions

```

QString getColumnnameForIndex (int index)
void _adaptColumnWidth (int index)

```

### Private Members

```

bool _skip_slot_sectionResized = false

```

### Private Slots

```

void slot_sectionResized (int index, int oldsize, int newsize)

```

### Private Static Attributes

```

std::shared_ptr<sh::configuration::ConfigurationValue> cfgvalFileListIconSize = sh::configuration::ConfigurationMan

```

## 10.1.397 Class sh::ui::qt::QtFilePropertyDialog

```

class sh::ui::qt::QtFilePropertyDialog : public sh::ui::qt::QtDialog, public sh::ui::FilePropertyDialog
    Qt based properties dialog.

```

## Public Functions

**~QtFilePropertyDialog()**

**QtFilePropertyDialog** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> *nodes*)

Constructed only by the infrastructure and made available otherwise.

void **init** (std::shared\_ptr<*Dialog*> *dialog*) **override**

Executes custom stuff on initialization, e.g. for populating the dialog.

void **refresh** () **override**

Refreshes the dialog content.

*sh::ui::FilePropertyDialogTabTableView* \***createTabViewTable** () **override**

Creates a new tab view table sub-widget.

*sh::ui::FilePropertyDialogTabTextView* \***createTabViewText** () **override**

Creates a new tab view text sub-widget.

*sh::ui::FilePropertyDialogTabIconTextBannerView* \***createTabViewIconTextBanner** () **override**

Creates a new tab view icon text banner sub-widget.

QList<std::shared\_ptr<*FilePropertyDialogTab*>> **tabs** ()

Returns a list of all tabs.

*sh::ui::FilePropertyDialogTabActionsView* \***widgetAt** (std::shared\_ptr<*FilePropertyDialogTab*> *tab*,  
int *i*)

Returns the widget from a tab at a certain index.

int **widgetCount** (std::shared\_ptr<*FilePropertyDialogTab*> *tab*)

Returns the number of widgets in a tab.

qint64 **dialogId** ()

Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

bool **isInited** ()

Returns if this dialog is initialized.

Must be called in main thread.

bool **wasAccepted** ()

Returns if this dialog was ‘accepted’ by the user (typically by clicking some ‘OK’ button).

Must be called in main thread.

void **waitClosed** ()

Wait until the user closed the dialog in some way.

May be called in any thread.

void **close** ()

Closes the dialog.

Must be called in main thread.

bool **wasClosed** ()

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \*manager ()

Returns the *DialogManager* which hosts this dialog.

## Public Slots

void on\_btnClose\_clicked ()

void on\_btnRefresh\_clicked ()

void on\_tabWidget\_currentChanged (int index)

void selectTabByScrollPosition ()

## Public Static Functions

void addTabFactory (int i, std::shared\_ptr<*FilePropertyDialogTabFactory*> factory)

Adds a *FilePropertyDialogTabFactory* so the Properties dialogs show its content.

See also the REGISTER\_FILEPROPERTYDIALOGTAB macro.

## Parameters

- i: An index which controls the display order. Use one of the REGISTER\_FILEPROPERTYDIALOGTAB\_INDEX\_\* values in *FilePropertyDialogTabFactory* as base values.

## Private Members

*Ui::QtFilePropertyDialog* \*ui

QList<QWidget\*> \_internaltabwidgets

QList<QString> \_tabheads

int \_skip\_on\_tabWidget\_currentChanged = 0

int \_skip\_selectTabByScrollPosition = 0

QVBoxLayout \*scrollAreaLayout

## Friends

friend class FilePropertyDialogTab

### 10.1.398 Class sh::ui::qt::QtFilePropertyDialogTabActionsView

class sh::ui::qt::QtFilePropertyDialogTabActionsView: public QWidget, public sh::ui::FilePropertyDialogTab

A tab view which shows a main widget in the main part and some buttons below.

### Public Functions

**QtFilePropertyDialogTabActionsView** (QWidget *\*parent* = 0)

Is intended to be directly constructed from everywhere.

void **setContent** (*FilePropertyDialogTabViewContent* *\*w*)

Sets the main widget. .

void **setButtons** (QList<QString> *buttons*)

Sets the list of buttons. .

void **setVisible** (bool *v*)

Sets the view visible or hidden. .

*FilePropertyDialogTabViewContent* **\*content** ()

The main widget.

QList<QString> **buttons** ()

The list of buttons.

void **onButtonTriggered** (std::function<void> int *i*

> *fcn*, QObject *\*owner* = 0) Sets a handler for a click on one of the buttons (optionally bound to an owner lifetime).

## 10.1.399 Class sh::ui::qt::QtFilePropertyDialogTabIconTextBannerView

**class** *sh::ui::qt::QtFilePropertyDialogTabIconTextBannerView* : **public** QWidget, **public** *sh::ui::FileProp*

Qt based *FilePropertyDialogTabIconTextBannerView*.

### Public Functions

**QtFilePropertyDialogTabIconTextBannerView** (QWidget *\*parent* = 0)

void **clear** ()

Clears the contents.

void **insertIcon** (QIcon *icon*, int *sizePt* = 24)

Adds an icon.

void **insertText** (QString *text*)

Adds a text.

### Private Members

QHBoxLayout **\*\_layout**

### 10.1.400 Class `sh::ui::qt::QtFilePropertyDialogTabTableView`

**class** `sh::ui::qt::QtFilePropertyDialogTabTableView` : **public** `QTableView`, **public** `sh::ui::FilePropertyDialogTabTableView`  
 Qt based `FilePropertyDialogTabTableView`.

#### Public Types

**typedef** `QPair<int, int>` **ItemIndex**

#### Public Functions

**QtFilePropertyDialogTabTableView** (`QWidget *parent = 0`)

void **setContent** (`QList<QPair<QString, QString>> content`)  
 Sets the content.

`QList<ItemIndex>` **getSelectedItems** ()  
 Returns the selected cells.

`QVariant` **getItemValue** (`int r, int c`)  
 Returns a value of a cell.

#### Private Functions

void **createAndSetModel** ()

### 10.1.401 Class `sh::ui::qt::QtFilePropertyDialogTabTextView`

**class** `sh::ui::qt::QtFilePropertyDialogTabTextView` : **public** `QLabel`, **public** `sh::ui::FilePropertyDialogTabTextView`  
 Qt based `FilePropertyDialogTabTextView`.

#### Public Functions

**QtFilePropertyDialogTabTextView** (`QWidget *parent = 0`)

void **setContent** (`QString content`)  
 Sets the content.

### 10.1.402 Class `sh::ui::qt::QtFileView`

**class** `sh::ui::qt::QtFileView` : **public** `std::enable_shared_from_this<QtFileView>`  
 Abstract base class for views for the contents of one directory.

Subclassed by `sh::ui::qt::QtFileIconview`, `sh::ui::qt::QtFileList`

## Public Functions

**QtFileView** ()

Constructed only by the infrastructure and made available otherwise.

**~QtFileView** ()

void **setBackgroundColor** (QString *c*)

QString **backgroundColor** ()

QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> **selectedNodes** ()

void **setSelection** (const QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> *nodes*)

std::shared\_ptr<*sh::filesystem::FilesystemNode*> **node** ()

QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> **getAllVisibleNodes** ()

void **setSizeFormatting** (*sh::filesystem::SizeFormatting* *v*)

void **gotoDir** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> *n*)

void **setHiddenFilesVisible** (bool *value*)

QThread \***thread** ()

void **configure** (*sh::ui::qt::QtFileViewControl* \**viewctrl*)

void **focus** ()

void **setSort** (int *column*, Qt::SortOrder *order*)

QObject \***as\_qobject** ()

QWidget \***as\_qwidget** ()

QAbstractItemView \***as\_qabstractitemview** ()

*sh::ui::qt::QtFileViewControl* \***control** ()

## Private Members

*sh::ui::qt::QtFileViewControl* \*\_**viewctrl** = nullptr

QString **\_tmp\_bgColor**

QPoint **\_dragStartPosition**

int **\_async\_gotodir\_index** = 0

bool **\_dragIsValid** = false

int **\_sort\_column** = 0

Qt::SortOrder **\_sort\_order** = Qt::AscendingOrder

## Friends

```
friend class sh::actions::common::ActionHistoryNavigateForward
friend class sh::actions::common::ActionHistoryNavigateBackward
friend class sh::actions::common::ActionNavigateInHistory
friend class sh::ui::qt::QtFileViewControl
class EventFilter : public QObject
```

## Public Functions

```
EventFilter (QObject *parent, QtFileView *fileview, sh::ui::qt::QtFileViewControl *fileviewctrl)
bool eventFilter (QObject *object, QEvent *event)
```

## Private Members

```
QtFileView *fileview
sh::ui::qt::QtFileViewControl *fileviewctrl
```

### 10.1.403 Class sh::ui::qt::QtFileViewControl

```
class sh::ui::qt::QtFileViewControl : public sh::ui::FileView
    Qt based file view.
```

## Public Functions

```
QtFileViewControl (sh::ui::qt::QtFilesystemPanel *panel, int i)
~QtFileViewControl ()
sh::ui::ColumnDimensions *columnDimensions () override
    Returns the column dimensions handler. .
sh::tools::HistoryTracker<std::shared_ptr<const sh::filesystem::Eurl>> *historyTracker ()
    Returns the history tracker. .                                override
FileViewMode viewmode () override
    Returns the view mode (icons, list, ... ?). .
void setViewmode (FileViewMode m) override
    Sets the view mode. .
int index () override
    Returns the position of this file view within the panel. .
void gotoDir (std::shared_ptr<sh::filesystem::FilesystemNode> n) override
    Jumps to a new current directory.
    Implementations have to call the base class implementation inside.
sh::filesystem::SizeFormatting getSizeFormattingMode () override
    Returns the mode how file sizes are formatted for displaying. .
```

void **setSizeFormattingMode** (*sh::filesystem::SizeFormatting mode*) **override**  
Sets the mode how file sizes are formatted for displaying. .

bool **hiddenFilesVisible** () **override**  
Returns if hidden files are visible. .

void **setHiddenFilesVisible** (bool *v*) **override**  
Sets the visibility of hidden files. .

int **iconDimension** () **override**  
Returns the icon size (in pixels). .

void **setIconDimension** (int *v*) **override**  
Sets the icon size (in pixels). .

void **setSort** (int *column*, Qt::SortOrder *order*) **override**  
Sets how to sort this view. .

int **sortColumn** () **override**  
Returns the index of the current sort column. .

Qt::SortOrder **sortOrder** () **override**  
Returns the current sort order. .

QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> **selectedNodes** () **override**  
Returns the nodes which the user has selected in this fileview. .

void **setSelection** (const QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> *nodes*) **override**  
Sets the node selection of this fileview. .

QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> **getAllVisibleNodes** () **override**  
Returns a list of all nodes currently listed in this file view. .

void **emit\_nodesActivated** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> *nodes*)

void **emit\_selectionChanged** ()

std::shared\_ptr<*sh::filesystem::FilesystemNode*> **node** ()  
Returns the current directory.

void **reload** (bool *skipModel* = false)  
Reloads the data inside this file view.

*sh::filesystem::FilesystemModelFileviewProxy* \***filemodel** ()  
Returns the filesystem model for this view. .

void **setFilemodel** (*sh::filesystem::FilesystemModelFileviewProxy* \**model*)  
Sets the filesystem model for this view. .

## Signals

void **nodesActivated** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> *nodes*)

void **selectionChanged** ()  
Triggered when the selection have changed.

void **viewOptionChanged** ()  
Triggered when some view options have changed.



## Private Functions

void **setIndex** (int *i*)

## Private Members

*sh::ui::qt::QtFilesystemPanel* \*\_**panel**

int **i**

*sh::ui::ColumnDimensions* \*\_**columnndimensions**

*sh::tools::HistoryTracker*<std::shared\_ptr<const *sh::filesystem::Eurl*>> \*\_**historyTracker**

*FileViewMode* **\_viewmode** = *FileViewMode::ListView*

bool **\_hiddenfilesvisible** = false

int **\_icondimension** = 0

int **\_sortColumn**

Qt::SortOrder **\_sortOrder**

*sh::filesystem::SizeFormatting* **\_sizeformattingmode** = *sh::filesystem::SizeFormatting::SizeFormattingModePrefixes*

## Friends

**friend class** *sh::ui::qt::QtFilesystemPanel*

### 10.1.404 Class *sh::ui::qt::QtFileView::EventFilter*

**class** *sh::ui::qt::QtFileView::EventFilter* : public QObject

## Public Functions

**EventFilter** (QObject \**parent*, *QtFileView* \**fileview*, *sh::ui::qt::QtFileViewControl* \**fileviewctrl*)

bool **eventFilter** (QObject \**object*, QEvent \**event*)

## Private Members

*QtFileView* \***fileview**

*sh::ui::qt::QtFileViewControl* \***fileviewctrl**

### 10.1.405 Class `sh::ui::qt::QtFilesystemPanel`

**class** `sh::ui::qt::QtFilesystemPanel` : **public** `QWidget`  
A splitted horizontal panel of file views.

#### Public Functions

```
QtFilesystemPanel (QWidget *parent = 0)
~QtFilesystemPanel ()
void addView (bool reinitialize = false)
void removeView (int i)
sh::ui::FileView *currentView ()
int currentViewIndex ()
int viewsCount ()
sh::ui::FileView *view (int i)
void selectFolder (std::shared_ptr<sh::filesystem::FilesystemNode> folder, int index = -1)
QList<std::shared_ptr<sh::filesystem::FilesystemNode>> selectedNodes ()
void _emit_viewmodeChanged (int i)
void buildView (std::shared_ptr<sh::ui::qt::QtFileView> *list, sh::ui::qt::QtFileViewControl *viewctrl,
                bool isNew)
void setCustomWidget (int i, std::shared_ptr<QWidget> w)
std::shared_ptr<sh::ui::qt::QtFileView> viewwidget (int i)
```

#### Signals

```
void panelFolderActivated (std::shared_ptr<sh::filesystem::FilesystemNode> folder, bool real-
                           Switch)
void panelCurrentViewChanged ()
void panelViewmodeChanged (int i)
void panelViewOptionChanged (int i)
void panelViewCountChanged ()
void panelSelectionChanged ()
```

#### Private Members

```
Ui::QtFilesystemPanel *ui
std::shared_ptr<sh::ui::qt::QtFileView> activeView
QString activeColor
QString inactiveColor
QList<std::shared_ptr<sh::ui::qt::QtFileView>> _views
QList<QWidget*> _mainviews
```

```
QHash<QWidget*, std::shared_ptr<QWidget>> _customwidgets
QList<sh::ui::qt::QtFileViewControl*> _viewcontrols
bool _skip_eventfilter = false
```

## Friends

```
friend class ui::qt::QtMainWindow
friend class ui::FileView
```

### 10.1.406 Class sh::ui::qt::QtJumpBar

```
class sh::ui::qt::QtJumpBar : public QWidget
    Button bar for navigating to places with parent-buttons and an text entry.
```

## Public Functions

```
QtJumpBar (QWidget *parent = 0)
~QtJumpBar ()
void setButtonMode ()
void setTextMode ()
bool isTextMode ()
std::shared_ptr<sh::filesystem::FilesystemNode> node ()
void setNode (std::shared_ptr<sh::filesystem::FilesystemNode> node)
QSize sizeHint () const
```

## Signals

```
void nodeChanged ()
void eurlRequested (std::shared_ptr<const sh::filesystem::Eurl> eurl)
```

## Private Functions

```
void _buildButtons ()
```

## Private Members

```
std::shared_ptr<sh::filesystem::FilesystemNode> _node = 0
Ui::QtJumpBar *ui
bool _isTextMode
```

### Private Slots

```
void on_btnOk_clicked ()  
void on_btnAbort_clicked ()
```

## 10.1.407 Class `sh::ui::qt::QtLinkButton`

```
class sh::ui::qt::QtLinkButton : public QToolButton  
    QToolButton with different look.  
    Subclassed by sh::ui::qt::QtSearchPanelButton
```

### Public Functions

```
QtLinkButton (QWidget *parent = 0)  
void setSizeByFactor (int f)  
void setMenu (QStringList items, QString menuchangetxt = tr("(change)"))  
void setSelectedItem (int idx)
```

### Signals

```
void menuItemSelected (int idx)
```

### Private Members

```
QString _linkcolor  
QStringList _menuitems  
QString _menuchangetxt
```

## 10.1.408 Class `sh::ui::qt::QtLogViewDialog`

```
class sh::ui::qt::QtLogViewDialog : public sh::ui::qt::QtDialog, public sh::ui::LogViewDialog  
    Qt based log view dialog.
```

### Public Functions

```
QtLogViewDialog (QString headtext, QString subheadtxt, base::LogSeverity minseverity)  
~QtLogViewDialog ()  
QString headtext ()  
    Returns the header text.  
QString subheadtxt ()  
    Returns the 2nd header text.  
sh::base::LogSeverity minimumSeverity ()  
    Returns the minimum severity.
```

qint64 **dialogId** ()  
Returns the dialog id.  
Each instance has an id unique in the complete Shallot process lifetime.  
Must be called in main thread.

bool **isInited** ()  
Returns if this dialog is initialized.  
Must be called in main thread.

bool **wasAccepted** ()  
Returns if this dialog was 'accepted' by the user (typically by clicking some 'OK' button).  
Must be called in main thread.

void **waitClosed** ()  
Wait until the user closed the dialog in some way.  
May be called in any thread.

void **close** ()  
Closes the dialog.  
Must be called in main thread.

bool **wasClosed** ()  
Returns if this dialog was closed.  
Must be called in main thread.

*DialogManager* \***manager** ()  
Returns the *DialogManager* which hosts this dialog.

### Private Functions

void **\_reloadLog** ()  
void **setMinimumSeverity** (*sh::base::LogSeverity* minseverity)

### Private Members

*Ui::QtLogViewDialog* \***ui**

### Private Slots

void **on\_btnClose\_clicked** ()  
void **on\_btnSaveToFile\_clicked** ()  
void **on\_btnSeverity\_clicked** ()

### 10.1.409 Class `sh::ui::qt::QtMainWindow`

**class** `sh::ui::qt::QtMainWindow` : **public** QMainWindow, **public** `sh::ui::MainWindow`  
The qt main window implementation.

#### Public Functions

**QtMainWindow** (QWidget \*parent = 0)

**~QtMainWindow** ()

**void initialize** () **override**  
Initializes this main window. .

**std::shared\_ptr<DialogManager> dialogManager** () **override**  
Returns the *DialogManager* which creates and drives *Dialogs* for this main window.  
You typically should not need to use it directly.

**int fileViewCount** () **override**  
Returns the current number of file views. .

**void addFileView** (bool *reinitialize* = false) **override**  
Adds a new file view. .

**void removeFileView** (int *i*) **override**  
Removes a file view. .

`sh::ui::FileView` \***currentFileView** () **override**  
Returns the file view which is currently active (i.e. focussed). .

`sh::ui::FileView` \***getFileView** (int *i*) **override**  
Returns a file view by position index. .

**void jumpToNode** (std::shared\_ptr<`sh::filesystem::FileSystemNode`> *node*) **override**  
Let the current view jump to another location. .

**void setTitlePattern** (QString *value*) **override**  
Sets the window title pattern. .

**void setTreeVisible** (bool *v*) **override**  
Sets the visibility of the directory tree. .

**void setFileDetailsPanelVisible** (bool *v*) **override**  
Sets the visibility of the file details panel. .

**std::shared\_ptr<`sh::filesystem::FileSystemNode`> currentDirectory** () **override**  
Gets the `sh::filesystem::FileSystemNode` selected in the current view. .

**std::shared\_ptr<`sh::ui::ActionExecutionInfoPanel`> addInfoPanel** (int *position*,  
`sh::actions::ActionExecutionInfo`  
\**info*, QColor *color* = QColor())  
**override**

Creates a new action execution panel.

Let this smart pointer die for removing it.

**void jumpbarSetTextMode** () **override**  
Sets the jumpbar to text input mode. .

**void jumpbarSetButtonMode** () **override**  
Sets the jumpbar to button mode. .

```

bool jumpbarIsTextMode () override
    Returns if the jumpbar is in text input mode. .

QString toolbarPosition () override
    Returns the current toolbar position. .

void setToolbarPosition (QString pos) override
    Sets the toolbar position. .

QString detailPanelPosition () override
    Returns the current detail panel position. .

void setDetailPanelPosition (QString pos) override
    Sets the detail panel position. .

std::shared_ptr<AboutDialog> showAboutDialog () override
    Shows and returns an 'About' dialog. .

std::shared_ptr<ManageProfilesDialog> showManageProfilesDialog () override
    Shows and returns a 'Manage Saved Settings' dialog. .

std::shared_ptr<OpenWithDialog> showOpenWithDialog (std::shared_ptr<sh::filesystem::FilesystemNode>
                                                    node, QList<std::shared_ptr<sh::tools::filetypes::OpenMethod>>
                                                    openMethods) override
    Shows and returns a 'Open With' dialog. .

std::shared_ptr<StoreProfileDialog> showStoreProfileDialog (std::shared_ptr<const
                                                            sh::filesystem::Eurl>      eurl)
                                                            override
    Shows and returns a 'Save Settings' dialog. .

std::shared_ptr<TuningDialog> showTuningDialog () override
    Shows and returns a 'Tuning' dialog. .

std::shared_ptr<LogViewDialog> showLogViewDialog (QString      headtext      =   QString(),
                                                    QString      subheadtxt    =   QString(),
                                                    sh::base::LogSeverity      minseverity =
                                                    sh::base::LogSeverity::_DEFAULT)
                                                    override
    Shows and returns a 'Log' dialog. .

std::shared_ptr<ManageBookmarksDialog> showManageBookmarksDialog () override
    Shows and returns a 'Manage Bookmarks' dialog. .

std::shared_ptr<FilePropertyDialog> showFilePropertyDialog (QList<std::shared_ptr<sh::filesystem::FilesystemNode>>
                                                            nodes) override
    Shows and returns a 'File Properties' dialog. .

std::shared_ptr<ActionExecutionInfoDialog> createActionExecutionInfoDialog (sh::actions::ActionExecutionInfo
                                                                              *info)
                                                                              override
    Creates an action execution dialog. .

void handleCloseAppRejected (QString rejectmsg) override
    React on a rejected application close request (with some feedback message). .

void handleClosed () override
    Do some cleanup steps just when closing starts. Implementations must call base implementation!

sh::ui::qt::QtUIStyle *uiStyle ()
    Gets a QtUIStyle instance, which helps for common ui styling.

void _initialize (sh::base::SingletonInitializer *singletonInitializer)
    Initializes the main window. For custom initialization logic, see MainWindow.initialize. .

```

void **fileViewsReloadAll** ()  
Reload all content in each file view.

void **onFileViewOptionsChanged** (std::function<void> int  
> *fcn*, QObject \**owner* = 0) Sets a handler for some view options in a file view changed (optionally bound to an owner lifetime).

void **onCurrentFileViewChanged** (std::function<void>  
> *fcn*) QObject \**owner* = 0) Sets a handler for the currently active file view changed (optionally bound to an owner lifetime).

void **onFileViewCountChanged** (std::function<void>  
> *fcn*) QObject \**owner* = 0) Sets a handler for the number of file views changed (optionally bound to an owner lifetime).

void **onCurrentDirectoryChanged** (std::function<void>  
> *fcn*) QObject \**owner* = 0) Sets a handler for the current directory in the active file view changed (optionally bound to an owner lifetime).

void **onGlobalViewOptionsChanged** (std::function<void>  
> *fcn*) QObject \**owner* = 0) Sets a handler for some global view options changed (optionally bound to an owner lifetime).

void **onCurrentProfileChanged** (std::function<void>  
> *fcn*) QObject \**owner* = 0) Sets a handler for the current profile changed (optionally bound to an owner lifetime).

void **jumpToEurl** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*)  
Let the current view jump to another location. .

QString **titlePattern** ()  
Returns the window title pattern.

void **setCurrentProfile** (QString *profile*)  
Sets the current profile. .

QString **currentProfile** ()  
Returns the current profile.

void **reloadProfile** ()  
Reloads the profile data.

bool **treeVisible** ()  
Returns the visibility of the directory tree.

void **setTreeSticky** (bool *v*)  
Sets if the directory tree should follow the active file view. .

bool **treeSticky** ()  
Returns if the directory tree follows the active file view.

bool **fileDetailsPanelVisible** ()  
Returns the visibility of the file details panel.

std::shared\_ptr<*sh::ui::ActionExecutionInfoPanel*> **addInfoPanel** (*sh::actions::ActionExecutionInfo*  
\**info*, QColor *color* = QColor())  
Creates a new action execution panel.  
Let this smart pointer die for removing it.

std::shared\_ptr<*ActionExecutionInfoPanel*> **showErrorPanel** ()  
Shows an error panel.



void **\_closeDirectly** ()  
 Directly begin application shutdown without checks.

bool **closeApp** ()  
 Check if the application may shut down now, and if so, initiate it.

bool **isCloseable** (QString \*msg = nullptr)  
 Check if the application may shut down now.

bool **isClosing** ()  
 Returns if the application is currently closing.

## Public Static Functions

bool **tryCreateMainWindow** (MainWindow \*&mainWindow)  
*QtMainWindow* \*mainWindow ()

void **setMainWindow** (MainWindow \*mainWindow)  
 Sets the main window instance. .

bool **isReady** ()  
 Returns if this process ui is ready (i.e. is created or runs headless).

bool **runsHeadless** ()  
 Returns if this process runs headless, i.e. without any ui, just for a background process.

void **\_closeDirectly** (sh::base::SingletonInitializer \*singletonInitializer)

QString **uiMode** ()  
 Returns the UI mode (e.g. qt, web).

## Private Functions

void **\_setupGlobalShortcut** (std::shared\_ptr<sh::actions::AbstractActionItem> action)

void **\_setupGlobalShortcut\_recursive** (std::shared\_ptr<sh::actions::SubmenuActionItem>  
 submenu)

void **\_addPermanentActionToToolbar** (std::shared\_ptr<sh::actions::SubmenuActionItem> a,  
 bool rightSide = false, bool preventDefaultAction =  
 false)

void **\_refreshToolbar** (std::shared\_ptr<sh::actions::ActionInstantiation>)

void **\_jumpToNode** (std::shared\_ptr<sh::filesystem::FilesystemNode> node, int skip)

void **\_jumpToIndex** (QModelIndex idx, int skip)

void **\_newToStatusbar\_helper** (sh::ui::qt::QtActionExecutionInfoPanel \*w)

void **\_refresh\_detailthumbnailvisibility** ()

void **\_thumbnail\_resized** ()

void **\_resizethumbnail** ()

void **\_detailpanel\_resized** ()

void **\_resizedetailpanel** ()

void **requestDetailThumbnail** (bool force = true)

void **setStatusbarVisibility** (bool v)

### Private Members

```
Ui::QtMainWindow *ui
sh::ui::qt::QtFilesystemPanel *fspanel
sh::filesystem::FilesystemModelDirectoryTreeProxy *treemodel = 0
sh::ui::qt::QtJumpBar *jumpbar
QTimer _statusbar_timer
int _statusbar_fullheight
int _statusbar_targetheight = 0
QSet<sh::ui::qt::QtActionExecutionInfoPanel*> _statusbar_childs
std::shared_ptr<sh::filesystem::FilesystemNode> _currentDirectory = 0
bool _request_detailthumbnail_skip = false
int _thumbnailwidth = 0
Qt::Orientation _myorientation = Qt::Horizontal
sh::ui::qt::QtUIStyle *_uistyle = 0
QAction *_toolbarLeftRightSplitSeparator
QHash<QString, std::shared_ptr<sh::actions::common::ActionGroups>> actionGroupsActions
QList<QtToolBarButtonHandler*> toolbarButtonHandlers
std::shared_ptr<QtDialogManager> _dialogManager
```

### Private Slots

```
void slot_detailbar_moved ()
void slot_toolbar_moved ()
void slot_statusbartimer ()
void slot_treeview_collapsedexpanded (const QModelIndex &index)
```

### Private Static Attributes

```
QtMainWindow *_qtMainWindow = nullptr
```

### Friends

```
friend class QtToolBarButton
friend class QtFilesystemPanel
class MyFlexibleLabel : public QLabel
    Needed internally in main window in order to have a label which really does not request any sizes.
```

## Public Functions

**MyFlexibleLabel** (QWidget \*parent = 0)

QSize **sizeHint** () **const override**

### 10.1.410 Class sh::ui::qt::QtMainWindow::MyFlexibleLabel

**class** *sh::ui::qt::QtMainWindow::MyFlexibleLabel* : **public** QLabel

Needed internally in main window in order to have a label which really does not request any sizes.

## Public Functions

**MyFlexibleLabel** (QWidget \*parent = 0)

QSize **sizeHint** () **const override**

### 10.1.411 Class sh::ui::qt::QtManageBookmarksDialog

**class** *sh::ui::qt::QtManageBookmarksDialog* : **public** *sh::ui::qt::QtDialog*, **public** *sh::ui::ManageBookmarksDialog*

Qt based manage bookmarks dialog.

## Public Functions

**QtManageBookmarksDialog** ()

**~QtManageBookmarksDialog** ()

qint64 **dialogId** ()

Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

bool **isInited** ()

Returns if this dialog is initialized.

Must be called in main thread.

bool **wasAccepted** ()

Returns if this dialog was 'accepted' by the user (typically by clicking some 'OK' button).

Must be called in main thread.

void **waitClosed** ()

Wait until the user closed the dialog in some way.

May be called in any thread.

void **close** ()

Closes the dialog.

Must be called in main thread.

bool **wasClosed** ()

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \*manager ()

Returns the *DialogManager* which hosts this dialog.

### Private Functions

void readBookmarks ()

void \_refresh\_values ()

void \_selectbookmark (QString id)

void \_selectfolder (QStringList name)

### Private Members

Ui::QtManageBookmarksDialog \*ui

### Private Slots

void on\_btnClose\_clicked ()

void on\_treeWidget\_itemSelectionChanged ()

void on\_btnDelete\_clicked ()

void on\_edtLabel\_textChanged (const QString &arg1)

void on\_edtLocation\_textChanged (const QString &arg1)

void on\_btnStore\_clicked ()

void on\_btnNew\_clicked ()

void on\_btnUp\_clicked ()

void on\_btnDown\_clicked ()

void on\_btnMove\_clicked ()

## 10.1.412 Class sh::ui::qt::QtManageProfilesDialog

**class** sh::ui::qt::QtManageProfilesDialog: public sh::ui::qt::QtDialog, public sh::ui::ManageProfilesDialog  
Qt based manage saved settings dialog.

### Public Functions

QtManageProfilesDialog ()

~QtManageProfilesDialog ()

qint64 dialogId ()

Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

bool **isInitiated** ()  
Returns if this dialog is initialized.  
Must be called in main thread.

bool **wasAccepted** ()  
Returns if this dialog was ‘accepted’ by the user (typically by clicking some ‘OK’ button).  
Must be called in main thread.

void **waitClosed** ()  
Wait until the user closed the dialog in some way.  
May be called in any thread.

void **close** ()  
Closes the dialog.  
Must be called in main thread.

bool **wasClosed** ()  
Returns if this dialog was closed.  
Must be called in main thread.

*DialogManager* \***manager** ()  
Returns the *DialogManager* which hosts this dialog.

### Private Functions

void **\_createProfileList** ()  
void **\_createSettingsList** ()  
void **\_createNodesList** ()

### Private Members

*Ui::QtManageProfilesDialog* \***ui**  
QStringListModel \***\_model**  
*sh::settings::ProfileNode* \***\_currentNode** = 0  
QHash<QListWidgetItem\*, QString> **\_nodes**

### Private Slots

void **on\_comboBox\_currentTextChanged** (const QString &*arg1*)  
void **on\_btnClose\_clicked** ()  
void **on\_btnDelNode\_clicked** ()  
void **on\_btnDelProfile\_clicked** ()  
void **slot\_listViewactivated** (QListWidgetItem\*, QListWidgetItem\*)

### 10.1.413 Class `sh::ui::qt::QtOpenWithDialog`

**class** `sh::ui::qt::QtOpenWithDialog` : **public** `sh::ui::qt::QtDialog`, **public** `sh::ui::OpenWithDialog`  
Qt based open with dialog.

#### Public Functions

**QtOpenWithDialog** (std::shared\_ptr<`sh::filesystem::FilesystemNode`> *node*,  
QList<std::shared\_ptr<`sh::tools::filetypes::OpenMethod`>> *openMethods*)

**~QtOpenWithDialog** ()

void **setProgramList** (QList<std::shared\_ptr<`sh::tools::filetypes::OpenMethod`>> *openMethods*)

std::shared\_ptr<`sh::tools::filetypes::OpenMethod`> **chosenMethod** () **override**  
Returns the chosen open-method (program for opening the file).

bool **rememberForMimetype** () **override**  
Returns if the chosen method is checked to be remembered for the same mime-type in the future.

std::shared\_ptr<const `sh::filesystem::Eurl`> **rememberForDirectory** () **override**  
If it's checked for the chosen method to be remembered for some subdirectory in the future, it returns the Eurl of this subdirectory, otherwise `nullptr`.

bool **rememberForFile** () **override**  
Returns if the chosen method is checked to be remembered for the same file in the future.

std::shared\_ptr<`sh::filesystem::FilesystemNode`> **node** ()  
Returns the file node which is later to be opened.

QList<std::shared\_ptr<`tools::filetypes::OpenMethod`>> **openMethods** ()  
Returns the open-methods (programs for opening the file) to present for choice.

qint64 **dialogId** ()  
Returns the dialog id.  
  
Each instance has an id unique in the complete Shallot process lifetime.  
  
Must be called in main thread.

bool **isInited** ()  
Returns if this dialog is initialized.  
  
Must be called in main thread.

bool **wasAccepted** ()  
Returns if this dialog was 'accepted' by the user (typically by clicking some 'OK' button).  
  
Must be called in main thread.

void **waitClosed** ()  
Wait until the user closed the dialog in some way.  
  
May be called in any thread.

void **close** ()  
Closes the dialog.  
  
Must be called in main thread.

bool **wasClosed** ()  
Returns if this dialog was closed.  
  
Must be called in main thread.

*DialogManager* \*manager ( )

Returns the *DialogManager* which hosts this dialog.

### Private Members

std::shared\_ptr<sh::tools::filetypes::OpenMethod> \_chosenMethod

std::shared\_ptr<const sh::filesystem::Eurl> \_rememberfordirectory

QAbstractItemModel \*\_listmodel

Ui::QtOpenWithDialog \*ui

### Private Slots

void on\_btnCancel\_clicked ( )

void on\_btnOk\_clicked ( )

void on\_btnSelectOther\_clicked ( )

void on\_listView\_activated (const QModelIndex &index)

void on\_btnAnotherAncestor\_clicked ( )

void on\_chkRememberDir\_toggled (bool checked)

## 10.1.414 Class sh::ui::qt::QtOpenWithDialogModel

**class** sh::ui::qt::QtOpenWithDialogModel : public QStringListModel

Used internally in OpenWithDialogModel.

### Public Functions

QtOpenWithDialogModel (QList<std::shared\_ptr<sh::tools::filetypes::OpenMethod>> methods,  
QObject \*parent = 0)

QVariant data (const QModelIndex &index, int role) const

### Private Members

QList<std::shared\_ptr<sh::tools::filetypes::OpenMethod>> \_methods

## 10.1.415 Class sh::ui::qt::QtSearchPanel

**class** sh::ui::qt::QtSearchPanel : public QWidget

Search panel for usage in the Qt main window.

### Public Functions

```
QtSearchPanel (std::shared_ptr<sh::filesystem::FilesystemNode> node, QWidget *parent = 0)
~QtSearchPanel ()
```

### Friends

```
friend class QtSearchPanelConfiguration
```

## 10.1.416 Class sh::ui::qt::QtSearchPanelAbstractEditor

```
template<class T1, class T2>
class sh::ui::qt::QtSearchPanelAbstractEditor : public T1, public T2
    Helper for other Qt based search panel editors.
```

### Public Functions

```
QtSearchPanelAbstractEditor (QWidget *parent = 0)
bool isEnabled ()
void setEnabled (bool v)
```

## 10.1.417 Class sh::ui::qt::QtSearchPanelButton

```
class sh::ui::qt::QtSearchPanelButton : public sh::ui::qt::QtLinkButton, public sh::ui::SearchPanelButton
    Qt based SearchPanelButton.
```

### Public Functions

```
QtSearchPanelButton (QWidget *parent = 0)
void setButtonText (QString txt)
    Sets the button text. .
void setMenuSelection (int i)
    Sets the currently selected menu item (for buttons with menus). .
void setSizeByFactor (int f)
void setMenu (QStringList items, QString menuchange = tr("(change)")
void setSelectedMenuItem (int idx)
```



## Signals

void **menuItemSelected** (int *idx*)

### 10.1.418 Class `sh::ui::qt::QtSearchPanelConfiguration`

**class** `sh::ui::qt::QtSearchPanelConfiguration` : **public** `sh::ui::SearchPanelConfiguration`  
 Search panel configuration for usage in a Qt ui.

#### Public Functions

**QtSearchPanelConfiguration** (*QtSearchPanel* \**owner*, QHBoxLayout \**editors*)

*SearchPanelButton* \***addMenuButton** (QString *text*, QStringList *menu*, std::function<void> int  
 > *onchanged*) Adds and returns a button for a menu. .

*SearchPanelButton* \***addActionButton** (QString *text*, std::function<void>  
 > *action*) Adds and returns a button for an action. .

*SearchPanelTextEditor* \***addTextEditor** ()  
 Adds and returns a text editor. .

*SearchPanelDateTimeEditor* \***addDateTimeEditor** ()  
 Adds and returns a date/time editor. .

*SearchPanelLabelEditor* \***addLabel** ()  
 Adds and returns a label. .

*SearchPanelSpacerEditor* \***addSpacer** ()  
 Adds and returns a spacer. .

*SearchPanelAbstractEditor* \***getEditorAt** (int *i*)  
 Returns the editor widget at a given position. .

void **onDestroyed** (std::function<void>  
 > *fcn*) *QObject* \**owner* = 0 Sets a handler for panel removal (optionally bound to an owner lifetime).

#### Public Members

*\_SearchPanelConfiguration\_HelperQObject* **myqobject**

#### Private Members

*QtSearchPanel* \***owner**

QHBoxLayout \***editors**

### 10.1.419 Class `sh::ui::qt::QtSearchPanelDateTimeEditor`

**class** `sh::ui::qt::QtSearchPanelDateTimeEditor` : public `sh::ui::qt::QtSearchPanelAbstractEditor`<QDateTimeEditor>  
Qt based `SearchPanelDateTimeEditor`.

#### Public Functions

**QtSearchPanelDateTimeEditor** (QWidget *\*parent* = 0)

QDateTime **datetime** ()  
Returns the selected date/time. .

void **setDatetime** (QDateTime *s*)  
Sets the selected date/time.

bool **isEnabled** ()  
void **setEnabled** (bool *v*)

### 10.1.420 Class `sh::ui::qt::QtSearchPanelLabelEditor`

**class** `sh::ui::qt::QtSearchPanelLabelEditor` : public `sh::ui::qt::QtSearchPanelAbstractEditor`<QLabel, `sh::ui::SearchPanelLabelEditor`>  
Qt based `SearchPanelLabelEditor`.

#### Public Functions

**QtSearchPanelLabelEditor** (QWidget *\*parent* = 0)

QString **textContent** ()  
Returns the text content. .

void **setTextContent** (QString *s*)  
Sets the text content. .

`SearchPanelAbstractEditor` **\*focusProxyEditor** ()  
Returns the focus proxy widget (which focus gets redirected to in some situations). .

void **setFocusProxyEditor** (`SearchPanelAbstractEditor` *\*w*)  
Sets the focus proxy widget (which focus gets redirected to in some situations). .

bool **isEnabled** ()  
void **setEnabled** (bool *v*)

### 10.1.421 Class `sh::ui::qt::QtSearchPanelSpacerEditor`

**class** `sh::ui::qt::QtSearchPanelSpacerEditor` : public `sh::ui::qt::QtSearchPanelAbstractEditor`<QLabel, `sh::ui::SearchPanelSpacerEditor`>  
Qt based `SearchPanelSpacerEditor`.

## Public Functions

```
QtSearchPanelSpacerEditor (QWidget *parent = 0)
bool isEnabled ()
void setEnabled (bool v)
```

### 10.1.422 Class sh::ui::qt::QtSearchPanelTextEditor

```
class sh::ui::qt::QtSearchPanelTextEditor : public sh::ui::qt::QtSearchPanelAbstractEditor<QLineEdit, sh::ui::Se
Qt based SearchPanelTextEditor.
```

## Public Functions

```
QtSearchPanelTextEditor (QWidget *parent = 0)
QString textContent ()
    Returns the text content. .
void setTextContent (QString s)
    Sets the text content. .
QString placeholderDescription ()
    Returns the placeholder description text (visible if field is empty). .
void setPlaceholderDescription (QString s)
    Sets the placeholder description text (visible if field is empty). .
bool isEnabled ()
void setEnabled (bool v)
```

### 10.1.423 Class sh::ui::qt::QtSettingUIFrame

```
class sh::ui::qt::QtSettingUIFrame : public QFrame
    User interface for one handling one setting.
```

## Public Functions

```
QtSettingUIFrame (sh::settings::Setting *setting, bool withcheckbox, QString displayvalue, QString
    additionalCheck, QVariant additionalCheckValue, QWidget *parent = 0)
bool isChecked ()
void setChecked (bool val)
void setAdditionalCheckVisible (bool v)
QVariant additionalCheckValue ()
```

### Private Members

```
QCheckBox *_checkbox = 0
QCheckBox *_additionalcheckbox = 0
QLabel *_additionalcheckboxlabel = 0
QWidget *_additionalcheckwidget = 0
QVariant _additionalCheckValue
```

## 10.1.424 Class `sh::ui::qt::QtStoreProfileDialog`

```
class sh::ui::qt::QtStoreProfileDialog : public sh::ui::qt::QtDialog, public sh::ui::StoreProfileDialog
    Qt based save settings dialog.
```

### Public Functions

```
QtStoreProfileDialog (std::shared_ptr<const sh::filesystem::Eurl> eurl)
~QtStoreProfileDialog ()

QList<sh::ui::StoreProfileDialog::SettingEntry> checkedSettings () override
    Returns the list of settings the user has checked to store.

QString profileName () override
    Returns the profile the user has chosen to store settings for.

bool withSubDirectories () override
    Returns if the user has chosen to apply the checked settings also for subdirectories.

QStringList inheritsFrom () override
    Returns the list of profiles the user has chosen to inherit from.

bool hasGlobal () override
    Returns if the user has chosen to apply the checked settings for each directory (instead of the current
    directory).

std::shared_ptr<const sh::filesystem::Eurl> eurl ()
    Returns the current directory this dialog shall work on.

qint64 dialogId ()
    Returns the dialog id.

    Each instance has an id unique in the complete Shallot process lifetime.

    Must be called in main thread.

bool isInited ()
    Returns if this dialog is initialized.

    Must be called in main thread.

bool wasAccepted ()
    Returns if this dialog was ‘accepted’ by the user (typically by clicking some ‘OK’ button).

    Must be called in main thread.
```

void **waitClosed** ()  
 Wait until the user closed the dialog in some way.  
 May be called in any thread.

void **close** ()  
 Closes the dialog.  
 Must be called in main thread.

bool **wasClosed** ()  
 Returns if this dialog was closed.  
 Must be called in main thread.

*DialogManager* \***manager** ()  
 Returns the *DialogManager* which hosts this dialog.

### Private Functions

void **\_settingsLevel** (int *level*, bool *persist* = true)  
 void **\_setglobal** (bool *v*)

### Private Members

QHash<*sh::ui::qt::QtSettingUIFrame*\*, std::shared\_ptr<*sh::settings::Setting*>> **\_widgets**  
 QHash<int, QLabel\*> **\_headerlabels**  
 QStringList **\_inheritProfileNameList**  
 bool **\_global** = false  
*Ui::QtStoreProfileDialog* \***ui**  
 int **\_level**

### Private Slots

void **on\_btnAddProf\_clicked** ()  
 void **on\_btnGlobal\_clicked** ()  
 void **on\_btn22\_clicked** ()  
 void **on\_toolButton\_2\_clicked** ()  
 void **on\_toolButton\_clicked** ()

### 10.1.425 Class `sh::ui::qt::QtToolBarButton`

**class** `sh::ui::qt::QtToolBarButton` : **public** `QToolBarButton`  
A qt toolbar button implementation with optional submenu.

#### Public Types

```
enum Position  
    Values:  
    enumerator LEFT  
    enumerator RIGHT  
    enumerator TOP  
    enumerator BOTTOM
```

#### Public Functions

```
QtToolBarButton (QWidget *parent = 0)  
void setText (const QString &text)  
QString text () const  
void setIcon (const QIcon &icon)  
QIcon icon ()  
QSize sizeHint () const  
void setButtonText (QString v)  
QString buttonText ()  
void setButtonIcon (QIcon v)  
QIcon buttonIcon ()  
void setButtonEnabled (bool v)  
bool isButtonEnabled ()  
bool hasExpander ()  
void setSubmenu (QtActionMenu *menu)  
QtActionMenu *submenu ()  
void setLocation (Position location)  
void setClickAction (std::function<void>  
    > action)  
void unsetClickAction ()  
void trigger ()  
void openMenu ()
```

## Private Functions

```
void _calcDims ()
void computeArrowAndDividerLineCoordinates ()
```

## Private Members

```
QString _text
QString _text1
QString _textWithoutMnemonic1
QString _text2
QString _textWithoutMnemonic2
QIcon _icon
bool _drawIconOnly = false
bool _hovered = false
bool _arrowhovered = false
sh::ui::qt::QtActionMenu *_menu = 0
std::function<void ()> _clickAction
Position location
int tw1
int tworig1
int tw2
int tworig2
int twmax
int tworigmax
int th
int thorig
int iw
int ih
int iedim
int ew
int eh
const int textpad = 20
const int opad = 3
int expx1
int expx2
int expy1
int expy2
```

```
int xt
int yt1
int yt2
int xi
int yi
QPainterPath arrow
bool _drawmnemonics = false
```

### Private Slots

```
void slot_clicked()
```

### Friends

```
friend class sh::actions::ActionsManager
```

## 10.1.426 Class sh::ui::qt::QtToolBarButtonHandler

```
class sh::ui::qt::QtToolBarButtonHandler
```

A handler which reflects the *sh::actions* objects to a graphical toolbar button (and keeps that up-to-date).

### Public Functions

```
QtToolBarButtonHandler (std::shared_ptr<sh::actions::SubmenuItem> action, bool preventDefaultAction, QtToolBarButton *button, QAction *qaction)
~QtToolBarButtonHandler ()
```

### Private Functions

```
void _updateSubmenu ()
```

### Private Members

```
std::shared_ptr<sh::actions::SubmenuItem> action
QtActionMenu menu
bool preventDefaultAction
QtToolBarButton *button
```



## Private Static Functions

```
void _applyPropertiesToButton (sh::actions::SubmenuItem *action, QtToolBarButton
                             *button, QAction *qaction)
```

### 10.1.427 Class sh::ui::qt::QtTuningDialog

```
class sh::ui::qt::QtTuningDialog : public sh::ui::qt::QtDialog, public sh::ui::TuningDialog
    Qt based tuning dialog.
```

## Public Functions

```
QtTuningDialog ()
```

```
~QtTuningDialog ()
```

```
qint64 dialogId ()
    Returns the dialog id.
```

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

```
bool isInitiated ()
    Returns if this dialog is initialized.
```

Must be called in main thread.

```
bool wasAccepted ()
    Returns if this dialog was 'accepted' by the user (typically by clicking some 'OK' button).
```

Must be called in main thread.

```
void waitClosed ()
    Wait until the user closed the dialog in some way.
```

May be called in any thread.

```
void close ()
    Closes the dialog.
```

Must be called in main thread.

```
bool wasClosed ()
    Returns if this dialog was closed.
```

Must be called in main thread.

```
DialogManager *manager ()
    Returns the DialogManager which hosts this dialog.
```

### Private Types

```
typedef QPair<QString, QWidget*> BoxWidgetByDesc
```

### Private Functions

```
QString _changedlglabel (std::shared_ptr<sh::configuration::ConfigurationValue> cv)  
void do_change (std::shared_ptr<sh::configuration::ConfigurationValue> cv)  
void filterVisibility (QString s)
```

### Private Members

```
Ui::QtTuningDialog *ui  
QList<BoxWidgetByDesc> boxWidgetsByDesc
```

### Private Slots

```
void on_btnCancel_clicked ()  
void on_lineEditSearchInt_textChanged (const QString &arg1)  
void on_lineEditSearchExt_textChanged (const QString &arg1)  
void on_lineEditSearchBehav_textChanged (const QString &arg1)  
void on_lineEditSearchAppear_textChanged (const QString &arg1)  
void on_tabWidget_currentChanged (int index)
```

## 10.1.428 Class *sh::ui::qt::QtUIStyle*

```
class sh::ui::qt::QtUIStyle
```

Methods for applying the Shallot visible style.

See *sh::ui::qt::QtMainWindow* about how to get an instance.

### Public Functions

```
QtUIStyle (sh::ui::qt::QtMainWindow *mainwnd)  
    Constructed only by the infrastructure and made available otherwise.  
QColor windowColor ()  
QColor backgroundColor ()  
QColor inactiveBackgroundColor ()  
QColor windowColorHighlighted ()  
QColor windowColorHalfHighlighted ()  
QColor windowColorDarkened ()  
QColor windowColorLikeTitlebar ()  
QColor linkColor ()
```

```

QColor titlebarColor ()
QColor foregroundColor ()
QColor foregroundColorLighter1 ()
QColor foregroundColorLighter2 ()
int fontSizeInPt (int r = 100)
QColor mix (float n1, QColor c1, QColor c2)
Sheet createSheet ()

```

## Private Members

```

sh::ui::qt::QtMainWindow * _mainWindow
class Sheet

```

## Public Functions

```

QString sheet ()
Sheet customcss (QString css)
Sheet fontsize_byFactor (int f)
Sheet fontsize_header ()
Sheet fontsize_smaller ()
Sheet textcolor (QColor color)
Sheet textcolor_lighter1 ()
Sheet textcolor_lighter2 ()
Sheet background (QColor color, bool restrictQWidget = true)
Sheet background_highlighted (bool restrictQWidget = true)
Sheet background_halfhighlighted (bool restrictQWidget = true)
Sheet background_darkened (bool restrictQWidget = true)
Sheet background_liketitlebar (bool restrictQWidget = true)
Sheet applyTo (QWidget *widget)
Sheet bold ()

```

## Private Functions

```

Sheet (QtUIStyle *style, QString sheet)

```

### Private Members

QString **\_sheet**  
*QtUIStyle* \***\_style**

### Friends

**friend class** QtUIStyle

## 10.1.429 Class `sh::ui::qt::QtUIStyle::Sheet`

**class** *sh::ui::qt::QtUIStyle::Sheet*

### Public Functions

QString **sheet** ()  
*Sheet* **customcss** (QString *css*)  
*Sheet* **fontsize\_byFactor** (int *f*)  
*Sheet* **fontsize\_header** ()  
*Sheet* **fontsize\_smaller** ()  
*Sheet* **textcolor** (QColor *color*)  
*Sheet* **textcolor\_lighter1** ()  
*Sheet* **textcolor\_lighter2** ()  
*Sheet* **background** (QColor *color*, bool *restrictQWidget* = true)  
*Sheet* **background\_highlighted** (bool *restrictQWidget* = true)  
*Sheet* **background\_halfhighlighted** (bool *restrictQWidget* = true)  
*Sheet* **background\_darkened** (bool *restrictQWidget* = true)  
*Sheet* **background\_liketitlebar** (bool *restrictQWidget* = true)  
*Sheet* **applyTo** (QWidget \**widget*)  
*Sheet* **bold** ()

### Private Functions

**Sheet** (*QtUIStyle* \**style*, QString *sheet*)

### Private Members

QString **\_sheet**

*QtUIStyle* \***\_style**

### Friends

**friend class** QtUIStyle

## 10.1.430 Class `sh::ui::qt::feedbackpanels::Credentials`

**class** *sh::ui::qt::feedbackpanels::Credentials* : **public** *sh::ui::qt::feedbackpanels::FeedbackPanel*  
*FeedbackPanel* for user login credentials (password based).

### Public Functions

**Credentials** (QString *domain*, QString *username*, QString *password*, QString *text*, bool *showDomain*,  
bool *showUsername*, bool *showPassword*, bool *showAnonymous*, bool *showRemember*)

**~Credentials** ()

void **cancelRequested** ()

bool **isClosed** ()

void **waitUntilClosed** ()

int **preferredHeight** ()

### Public Members

QString **domain**

QString **username**

QString **password**

bool **anonymous**

bool **remember**

bool **accepted**

### Signals

void **wasClosed** ()

### Private Members

*Ui::*Credentials \*ui

### Private Slots

void on\_chkAnonymous\_toggled (bool *checked*)

void on\_pushButton\_3\_clicked ()

void on\_pushButton\_4\_clicked ()

## 10.1.431 Class *sh::ui::qt::feedbackpanels::FeedbackPanel*

**class** *sh::ui::qt::feedbackpanels::FeedbackPanel* : **public** QWidget

Abstract base class for a Qt helper widget used in action user feedback.

Subclassed by *sh::ui::qt::feedbackpanels::Credentials*, *sh::ui::qt::feedbackpanels::GridForm*,  
*sh::ui::qt::feedbackpanels::MsgBox*, *sh::ui::qt::feedbackpanels::UnixPermissions*

### Public Functions

**FeedbackPanel** (QWidget \*parent = 0)

bool **isClosed** ()

void **waitUntilClosed** ()

int **preferredHeight** ()

void **cancelRequested** () = 0

### Signals

void **wasClosed** ()

### Private Members

QMutex **\_mutex**

QWaitCondition **\_closedcondition**

bool **\_closed** = false

## 10.1.432 Class *sh::ui::qt::feedbackpanels::GridForm*

**class** *sh::ui::qt::feedbackpanels::GridForm* : **public** *sh::ui::qt::feedbackpanels::FeedbackPanel*  
*FeedbackPanel* for grid based forms.

## Public Functions

```

GridForm (QString text, QList<sh::ui::qt::feedbackpanels::GridFormRow> form, QStringList answers,
            int defaultanswer = -1, int cancelanswer = -1, QWidget *parent = 0)
~GridForm ()
int preferredHeight ()
void cancelRequested ()
bool isClosed ()
void waitUntilClosed ()

```

## Public Members

```
int answer = -1
```

## Signals

```
void wasClosed ()
```

## Private Members

```

Ui::GridForm *ui
QHash<QPushButton*, int> btn2index
QPushButton *_cancelBtn = 0
QPushButton *_defaultBtn = 0

```

## Private Slots

```
void slot_btnclicked ()
```

### 10.1.433 Class *sh::ui::qt::feedbackpanels::GridFormInnerSimpleChooser*

```

class sh::ui::qt::feedbackpanels::GridFormInnerSimpleChooser : public QWidget
    Helper widget for a GridForm.

```

## Public Functions

```

GridFormInnerSimpleChooser (sh::actions::ActionExecutionUserFeedback::Choices *choices,
                             QWidget *parent = 0)
~GridFormInnerSimpleChooser ()

```

### Public Members

int **answer** = -1  
QString **text**

### Private Functions

bool **eventFilter** (QObject \*obj, QEvent \*event)

### Private Members

QLineEdit \***lineEdit**  
*sh::actions::ActionExecutionUserFeedback::Choices* \***choices**  
QVBoxLayout \***mylayout**  
*sh::actions::ActionExecutionUserFeedback::Choice* \***currentchoice** = 0

### Private Slots

void **slot\_chosen** (int id)  
void **slot\_textedited** (QString t)

## 10.1.434 Class *sh::ui::qt::feedbackpanels::MsgBox*

**class** *sh::ui::qt::feedbackpanels::MsgBox* : **public** *sh::ui::qt::feedbackpanels::FeedbackPanel*  
*FeedbackPanel* for showing a message, some answer buttons, and optionally a text input box.

### Public Functions

**MsgBox** (QString *question*, QList<QString> *answers*, QString *icon* = "", QString *withInputBox* =  
QString(), bool *inputBoxMultiline* = false, int *defaultanswer* = -1, int *cancelanswer* = -  
1, int *valuePreselectFrom* = -1, int *valuePreselectTo* = -1, QList<QString> *answericons* =  
QList<QString>(), QWidget \**parent* = 0)  
**~MsgBox** ()  
void **cancelRequested** ()  
bool **isClosed** ()  
void **waitUntilClosed** ()



## Public Members

int **answer** = -1

QString **text**

## Signals

void **wasClosed** ()

## Private Members

Ui::MsgBox \***ui**

QHash<QPushButton\*, int> **btn2index**

QWidget \***\_initialFocusWidget** = 0

QPushButton \***\_cancelBtn** = 0

QPushButton \***\_defaultBtn** = 0

bool **\_inputBoxMultiline**

int **\_valuePreselectFrom**

int **\_valuePreselectTo**

## Private Slots

void **slot\_btnclicked** ()

### 10.1.435 Class sh::ui::qt::feedbackpanels::UnixPermissions

**class** *sh::ui::qt::feedbackpanels::UnixPermissions* : **public** *sh::ui::qt::feedbackpanels::FeedbackPanel*  
*FeedbackPanel* for setting one set of unix filesystem permissions.

## Public Functions

**UnixPermissions** (QWidget \*parent = 0)

**~UnixPermissions** ()

void **cancelRequested** ()

void **setflags** (bool *userMayRead*, bool *userMayWrite*, bool *userMayExecute*, bool *groupMayRead*,  
bool *groupMayWrite*, bool *groupMayExecute*, bool *othersMayRead*, bool *othersMay-*  
*Write*, bool *othersMayExecute*, bool *sticky*, bool *setuid*, bool *setgid*)

void **setusers** (QStringList *users*, QString *selecteduser*)

void **setgroups** (QStringList *groups*, QString *selectedgroup*)

bool **isClosed** ()

void **waitUntilClosed** ()

int **preferredHeight** ()

### Public Members

bool **accepted**  
bool **userMayRead**  
bool **userMayWrite**  
bool **userMayExecute**  
bool **groupMayRead**  
bool **groupMayWrite**  
bool **groupMayExecute**  
bool **othersMayRead**  
bool **othersMayWrite**  
bool **othersMayExecute**  
bool **sticky**  
bool **setuid**  
bool **setgid**  
QString **user**  
QString **group**

### Signals

void **wasClosed** ()

### Private Members

*Ui::*UnixPermissions \***ui**

### Private Slots

void **on\_pushButton\_clicked** ()  
void **on\_pushButton\_2\_clicked** ()

## 10.1.436 Class *sh::ui::web::WebAboutDialog*

**class** *sh::ui::web::WebAboutDialog* : **public** QObject, **public** *sh::ui::web::WebDialog*, **public** *sh::ui::AboutDialog*  
Web based about dialog.

## Public Functions

### WebAboutDialog ()

QVariant **dialogProperty** (QString *dlgPropertyKey*, QVariant *deft* = QVariant())

Returns a dialog property value by key.

void **setDialogProperty** (QString *dlgPropertyKey*, QVariant *value*)

Sets a dialog property value for a key.

QVariantHash **dialogResult** ()

Returns the dialog result as hash.

The browser side of a web dialog can ‘accept’ a dialog while closing by setting a non-empty dialog result. This mechanism allows browser side dialogs to return data to the backend. It’s up to the backend to interpret the content of this hash.

If the dialog was cancelled, or is still open, the returned hash is empty. See also wasClosed().

void **triggerDialogEvent** (QString *name*, QJsonObject *data* = QJsonObject())

Triggers a dialog event (at the browsers).

They in turn typically fetch some data again in order to refresh the ui.

void **setCloseableByUser** (bool *v* = true)

Sets if this dialog shall be directly closeable by the user by the window decoration. Set to `false` if the user has to react on this dialog by clicking on some controls in the dialog body.

This method must be called during construction.

bool **isCloseableByUser** ()

Returns if this dialog shall be directly closeable by the user by the window decoration.

QJsonObject **toJson** () **override**

Returns the json representation as QJsonObject.

qint64 **dialogId** ()

Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

bool **isInited** ()

Returns if this dialog is initialized.

Must be called in main thread.

bool **wasAccepted** ()

Returns if this dialog was ‘accepted’ by the user (typically by clicking some ‘OK’ button).

Must be called in main thread.

void **waitClosed** ()

Wait until the user closed the dialog in some way.

May be called in any thread.

void **close** ()

Closes the dialog.

Must be called in main thread.

bool **wasClosed** ()

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \***manager** ( )

Returns the *DialogManager* which hosts this dialog.

### 10.1.437 Class sh::ui::web::WebActionExecutionInfoDialog

**class** *sh::ui::web::WebActionExecutionInfoDialog* : **public** *sh::ui::ActionExecutionInfoDialog*  
Web based action execution dialog.

#### Public Types

**enum** **MessageBoxButton**

Buttons in a message box from *ActionExecutionUserFeedback*.

*Values:*

**enumerator** **NONE** = 0

**enumerator** **OK** = 1 << 0

**enumerator** **Continue** = 1 << 1

**enumerator** **Cancel** = 1 << 2

**enumerator** **Retry** = 1 << 3

**enumerator** **Yes** = 1 << 4

**enumerator** **No** = 1 << 5

#### Public Functions

**WebActionExecutionInfoDialog** (*sh::actions::ActionExecutionInfo* \**info*)

void **setDetails** (QString *fv*, QString *fob*, QString *tv*, QString *tob*)

Sets the item details text ('from a/foo.jpg', 'to b/foo.jpg'). .

void **setHead** (QString *txt*)

Sets the header text. .

void **setProgress** (bool *isDeterminate*, quint64 *done*, quint64 *all*, QString *text*)

Sets the progress. .

int **messageBox** (QString *text*, QList<QString> *answers*, QString *icon* = QString(), int *defaultanswer* = -1, int *cancelanswer* = -1, QList<QString> *answericons* = QList<QString>())

int **inputBox** (QString *text*, QList<QString> *answers*, QString \**value*, QString *icon* = QString(), int *defaultanswer* = -1, int *cancelanswer* = -1, int *valuePreselectFrom* = -1, int *valuePreselectTo* = -1)

int **multilineInputBox** (QString *text*, QList<QString> *answers*, QString \**value*, QString *icon* = QString(), int *defaultanswer* = -1, int *cancelanswer* = -1)

int **simpleChooserGridform** (QString *text*, GridformEntries \**entries*, QStringList *answers*, int *defaultanswer* = -1, int *cancelanswer* = -1)

bool **credentialsDialog** (QString *text*, bool *showDomain*, bool *showUsername*, bool *showPassword*, bool *showAnonymous*, bool *showRemember*, QString \**domain*, QString \**username*, QString \**password*, bool \**isAnonymous*, bool \**isRemember*)

```

bool unixPermissionsDialog (bool *userMayRead, bool *userMayWrite, bool *userMayExecute,
                             bool *groupMayRead, bool *groupMayWrite, bool *groupMayExecute,
                             bool *othersMayRead, bool *othersMayWrite, bool *othersMayExecute,
                             bool *sticky, bool *setuid, bool *setgid, QStringList users,
                             QStringList groups, QString *ownerUser, QString *ownerGroup)

qint64 id ()
qint64 webts_created ()
QString details_fromverb ()
QString details_fromobject ()
QString details_toverb ()
QString details_toobject ()
QString head ()
bool progress_isDeterminate ()
qint64 progress_done ()
qint64 progress_all ()
QString progress_text ()
JsonValue userFeedbackAsJsonValue ()
bool isLogicallyVisible ()
    Returns if this dialog is logically visible (i.e. set visible by the action).
void setLogicallyVisible (bool v)
    Sets if this dialog is logically visible (i.e. set visible by the action). .
bool isBackground ()
    Returns if this dialog is currently in background mode (i.e. not visible).
void setBackground (bool v)
    Sets if this dialog is currently in background mode (i.e. not visible). .
void setForceForeground (bool v)
    Sets if this dialog is currently forced to be visible in foreground. .
QString simpleInputDialog (QString text, QString deflt, int valuePreselectFrom = -1, int valuePreselectTo = -1)
int simpleMessageBox (QString text, int buttons = (int)MessageBoxButton::OK, QString icon =
                     QString(), int defaultbutton = (MessageBoxButton)0, int cancelbutton =
                     (MessageBoxButton)0)

```

### Private Functions

```

void _handleUserFeedback (QString kind, std::shared_ptr<UserFeedback> userfeedback)
void _triggerChanged ()

```

### Private Members

```
qint64 _id
QString _details_fv
QString _details_fob
QString _details_tv
QString _details_tob
QString _head
bool _progress_isDeterminate
quint64 _progress_done
quint64 _progress_all
QString _progress_text
std::shared_ptr<UserFeedback> _currentUserFeedback = 0
QMutex _currentUserFeedbackMutex
QWaitCondition _currentUserFeedbackChangedCondition
qint64 _webts_created
```

### Private Static Functions

```
QByteArray iconToBase64Src (QString icon, int sizeInPt)
QList<QVariant> iconsToBase64Srcs (QStringList icons, int sizeInPt)
```

### Friends

```
friend class WebActionManager
class UserFeedback : public QMap<QString, QVariant>
```

### Public Members

```
qint64 id = -1
bool answered = false
```

## 10.1.438 Class sh::ui::web::WebActionExecutionInfoDialog::UserFeedback

```
class sh::ui::web::WebActionExecutionInfoDialog::UserFeedback : public QMap<QString, QVariant>
```

## Public Members

qint64 **id** = -1  
 bool **answered** = false

### 10.1.439 Class `sh::ui::web::WebActionExecutionInfoPanel`

**class** `sh::ui::web::WebActionExecutionInfoPanel` : **public** `sh::ui::ActionExecutionInfoPanel`  
 Web based action execution info panel.

## Public Functions

**WebActionExecutionInfoPanel** (`sh::actions::ActionExecutionInfo *info`, `QColor color`)  
**~WebActionExecutionInfoPanel** ()  
 void **setLabel** (`QString s`)  
     Sets the label text. .  
 void **setProgress** (`bool isProgressDeterminate`, `qint64 progressDone`, `qint64 progressAll`)  
     Sets the progress. .  
 void **setForceForeground** (`bool v`)  
     Sets if the associated action is currently forced to be visible in foreground. .  
 void **setPanelVisible** (`bool v`)  
     Sets if the panel is visible. .  
 void **setWidth** (`int width`)  
     Sets the panel with (in pixels). .  
 qint64 **id** ()  
 QString **label** ()  
 bool **isProgressDeterminate** ()  
 qint64 **progressDone** ()  
 qint64 **progressAll** ()  
 bool **forceForeground** ()  
 bool **panelVisible** ()  
 int **width** ()  
 QColor **color** ()  
`sh::actions::ActionExecutionInfo *info` ()  
 void **onClicked** (`std::function<void>`)  
     > `fcnQObject *owner` = 0 Sets a handler for a click on the button (optionally bound to an owner lifetime).  
 void **onDestroyed** (`std::function<void>`)  
     > `fcnQObject *owner` = 0 Sets a handler for panel removal (optionally bound to an owner lifetime).  
 void **onVisibilityChanged** (`std::function<void>`)  
     > `fcnQObject *owner` = 0 Sets a handler for panel visibility changes (optionally bound to an owner lifetime).

### Private Functions

```
void _triggerChanged ()
```

### Private Members

```
QColor _color
sh::actions::ActionExecutionInfo *_info
qint64 _id
QString _label
bool _isProgressDeterminate
quint64 _progressDone
quint64 _progressAll
bool _forceForeground
bool _panelVisible
int _width
```

### Friends

```
friend class WebActionManager
```

## 10.1.440 Class sh::ui::web::WebActionManager

**class** *sh::ui::web::WebActionManager* : public QObject, public *sh::ui::web::WebModule*  
Manages *sh::actions::AbstractActionItem* handling, e.g. showing them in the toolbar, executing them, and displaying their user interface.

### Public Functions

```
WebActionManager ()
```

```
void initialize ()
```

```
std::shared_ptr<WebActionExecutionInfoPanel> addInfoPanel (int position,  
                                                         sh::actions::ActionExecutionInfo  
                                                         *info, QColor color = QColor())
```

Adds and returns a new action info panel.

```
std::shared_ptr<WebActionExecutionInfoDialog> addActionExecutionInfoDialog (sh::actions::ActionExecutionInfo  
                                                                           *info)
```

Adds and returns a new action info dialog.

```
void refreshToolbar (std::shared_ptr<sh::actions::ActionInstantiation> instantiation)  
Refreshes the main toolbar.
```

```
bool rootCommand (WebServerEngineRequest *request, QString webadapter)  
Returns the root ('index.html') content. .
```

```
void setEngine (WebServerEngine *webserver)  
Assigns this web module to a sh::ui::web::WebServerEngine. .
```



*WebServerEngine* \*webserver ()

Returns the *sh::ui::web::WebServerEngine* this web module is assigned to (may be nullptr).

## Public Static Functions

bool **executeCommand\_byQObjectReflection** (QObject \*o, *WebServerEngineRequest* \*request, QString command, QString justLeftside = QString())

Convenience function often used by *executeCommand()*.

For a QObject, it searches there for a slot with the name cmd\_{command}, with `/'s replaced by '\_'s (so, for the command "foo/getBars", it searches for slotcmd\_foo\_getBars). If it finds one, it gets executed (in a worker thread). The request is considered as handled, if so.

If there is such a slot, but with appended \_\_M in name, it is called in main thread!

**Return** If it handled (answered) the command.

## Private Functions

bool **executeCommand** (*WebServerEngineRequest* \*request, QString command) **override**

Executes command as requested by the browser, if responsible.

Read parameters, check if this module is responsible for answering, and then write an answer with request.

Implementations often use *executeCommand\_byQObjectReflection()* for convenience.

**Return** If it handled (answered) the command.

void **\_observeToolBarAction** (*sh::actions::AbstractActionItem* \*a)

std::shared\_ptr<*sh::actions::AbstractActionItem*> **resolveActionpath** (QString sactionpath, QStringList \*actionpath, QList<QPair<QString, std::shared\_ptr<*sh::actions::AbstractActionItem*>>> \*subactions)

void **\_triggerActionuiChanged** ()

## Private Members

QList<*WebActionExecutionInfoPanel*\*> **\_infopanel**s

QList<*WebActionExecutionInfoDialog*\*> **\_infodialogs**

QTimer **\_triggerActionuiChangedTimer**

QTimer **\_triggerToolBarRefreshTimer**

QMap<QString, std::shared\_ptr<*sh::actions::SubmenuActionItem*>> **\_permanentToolBarActions**

### Private Slots

```
void cmd__answer_userfeedback__M(WebServerEngineRequest *request)
void cmd__execute(WebServerEngineRequest *request)
void cmd__get(WebServerEngineRequest *request)
void cmd__get_ui__M(WebServerEngineRequest *request)
void cmd__icon(WebServerEngineRequest *request)
void cmd__panel_clicked__M(WebServerEngineRequest *request)
```

### Private Static Attributes

```
QRegularExpression reActionTextAccel
```

### Friends

```
friend class WebActionExecutionInfoPanel
friend class WebActionExecutionInfoDialog
```

## 10.1.441 Class `sh::ui::web::WebDetailPanelManager`

**class** `sh::ui::web::WebDetailPanelManager` : **public** `QObject`, **public** `sh::ui::web::WebModule`  
Manages the detail panel (typically in bottom part of main window).

### Public Functions

**WebDetailPanelManager** ()

void **load** (`QList<std::shared_ptr<sh::filesystem::FilesystemNode>>` nodes)  
Loads details for a given list of nodes (typically called after they get selected).

bool **rootCommand** (`WebServerEngineRequest` \*request, `QString` webadapter)  
Returns the root ('index.html') content. .

void **setEngine** (`WebServerEngine` \*webserver)  
Assigns this web module to a `sh::ui::web::WebServerEngine`. .

`WebServerEngine` \***webserver** ()  
Returns the `sh::ui::web::WebServerEngine` this web module is assigned to (may be nullptr).

### Public Static Functions

bool **executeCommand\_byQObjectReflection** (`QObject` \*o, `WebServerEngineRequest` \*request, `QString` command, `QString` justLeftside = `QString()`)

Convenience function often used by `executeCommand()`.

For a `QObject`, it searches there for a slot with the name `cmd_{command}`, with ``/'`s replaced by `'_'`s (so, for the command "foo/getBars", it searches for `slotcmd_foo_getBars`). If it finds one, it gets executed (in a worker thread). The request is considered as handled, if so.

If there is such a slot, but with appended `__M` in name, it is called in main thread!

**Return** If it handled (answered) the command.

### Private Functions

bool **executeCommand** (*WebServerEngineRequest* \*request, QString command) **override**

Executes `command` as requested by the browser, if responsible.

Read parameters, check if this module is responsible for answering, and then write an answer with `request`.

Implementations often use *executeCommand\_byQObjectReflection()* for convenience.

**Return** If it handled (answered) the command.

### Private Members

QMutex **\_mutex**

QList<std::shared\_ptr<sh::paneldetails::PanelDetail>> **\_paneldetails**

qint64 **\_webts\_lastupdate**

### Private Slots

void **cmd\_\_get\_details** (*WebServerEngineRequest* \*request)

void **cmd\_\_get\_thumbnail** (*WebServerEngineRequest* \*request)

void **cmd\_\_link\_clicked** (*WebServerEngineRequest* \*request)

## 10.1.442 Class sh::ui::web::WebDialog

**class** *sh::ui::web::WebDialog* : **public** *sh::tools::Jsonable*

Abstract base class for a web based dialog. Typically used for also implementing some *sh::ui::Dialog*.

A web dialog is a dialog window presented to the user on browser side. A web dialog implementation consists of a backend part (a subclass of *WebDialog*) and a browser side part (basically a subclass of *shwebui.Dialog*).

For implementing a web dialog, implement a subclass of *WebDialog* (which in turn will specify the browser side part) and *sh::ui::Dialog*. Create such dialogs by means of *WebDialogManager*.

Communicating values between backend and browser typically works by *dialogProperty()* and *setDialogProperty()*, also *dialogResult()* for the dialog result after closing.

Subclassed by *sh::ui::web::WebAboutDialog*, *sh::ui::web::WebExceptionDialog*, *sh::ui::web::WebFilePropertyDialog*, *sh::ui::web::WebLogViewDialog*, *sh::ui::web::WebManageBookmarksDialog*, *sh::ui::web::WebManageProfilesDialog*, *sh::ui::web::WebOpenWithDialog*, *sh::ui::web::WebStoreProfileDialog*, *sh::ui::web::WebTuningDialog*

## Public Functions

**WebDialog** (QString *dialogClassName*)

See [WebDialogManager](#).

### Parameters

- **dialogClassName**: The name of the shwebui.Dialog subclass which implements the web dialog on browser side.

QVariant **dialogProperty** (QString *dlgPropertyKey*, QVariant *deflt* = QVariant())

Returns a dialog property value by key.

void **setDialogProperty** (QString *dlgPropertyKey*, QVariant *value*)

Sets a dialog property value for a key.

QVariantHash **dialogResult** ()

Returns the dialog result as hash.

The browser side of a web dialog can ‘accept’ a dialog while closing by setting a non-empty dialog result. This mechanism allows browser side dialogs to return data to the backend. It’s up to the backend to interpret the content of this hash.

If the dialog was cancelled, or is still open, the returned hash is empty. See also `wasClosed()`.

void **triggerDialogEvent** (QString *name*, QJsonObject *data* = QJsonObject())

Triggers a dialog event (at the browsers).

They in turn typically fetch some data again in order to refresh the ui.

void **setCloseableByUser** (bool *v* = true)

Sets if this dialog shall be directly closeable by the user by the window decoration. Set to `false` if the user has to react on this dialog by clicking on some controls in the dialog body.

This method must be called during construction.

bool **isCloseableByUser** ()

Returns if this dialog shall be directly closeable by the user by the window decoration.

**~WebDialog** ()

QJsonObject **toJson** () **override**

Returns the json representation as QJsonObject.

## Private Members

QVariantHash **\_properties**

QString **\_dialogClassName**

qint64 **\_webts\_created**

## Friends

```
friend class WebDialogManager
```

### 10.1.443 Class `sh::ui::web::WebDialogManager`

```
class sh::ui::web::WebDialogManager : public QObject, public sh::ui::web::WebModule, public sh::ui::DialogManager
    A DialogManager used in Web ui.
```

## Public Functions

```
WebDialogManager ()
```

```
WebServerEngine *webserver ()
```

Return the *WebServerEngine* hosting this dialog.

```
bool rootCommand (WebServerEngineRequest *request, QString webadapter)
```

Returns the root ('index.html') content. .

```
void setEngine (WebServerEngine *webserver)
```

Assigns this web module to a *sh::ui::web::WebServerEngine*. .

```
std::shared_ptr<Dialog> getDialogById (qint64 id)
```

Returns a *Dialog* by dialog id.

Must be called in main thread.

```
QList<std::shared_ptr<Dialog>> getAllDialogs ()
```

Returns a list of all dialogs currently shown (in order of creation).

Must be called in main thread.

```
QList<qint64> getAllDialogIds ()
```

Returns a list of dialog ids of all dialogs currently shown (in order of creation).

Must be called in main thread.

```
template<class TDlg, typename ...Args>
```

```
std::shared_ptr<TDlg> createAndShowDialog (Args... args)
```

Creates and shows a *Dialog* by class and constructor parameters.

Those dialogs (TDlg) have to implement *Dialog* (typically a subclass of it, representing a particular dialog, like *FilePropertyDialog*), and also some ui mode (e.g. qt, web) specific class (depends on that ui mode).

You typically should not have to use it outside of *MainWindow* or subclasses. The *MainWindow* interface allows to create all available dialogs in a ui mode independent way.

May be called in any thread.

**Return** The created *Dialog* instance.

## Public Static Functions

bool **executeCommand\_byQObjectReflection** (QObject \*o, *WebServerEngineRequest* \*request, QString command, QString justLeftside = QString())

Convenience function often used by *executeCommand()*.

For a QObject, it searches there for a slot with the name cmd\_{command}, with `/'s replaced by '\_'s (so, for the command "foo/getBars", it searches for slotcmd\_foo\_getBars). If it finds one, it gets executed (in a worker thread). The request is considered as handled, if so.

If there is such a slot, but with appended \_\_M in name, it is called in main thread!

**Return** If it handled (answered) the command.

## Private Functions

std::shared\_ptr<Dialog> **getDialogForRequest** (*WebServerEngineRequest* \*request)

Returns the *WebDialog* referred to the request (i.e. its dialogId parameter).

bool **executeCommand** (*WebServerEngineRequest* \*request, QString command) **override**

Executes command as requested by the browser, if responsible.

Read parameters, check if this module is responsible for answering, and then write an answer with request.

Implementations often use *executeCommand\_byQObjectReflection()* for convenience.

**Return** If it handled (answered) the command.

void **showDialog** (std::shared\_ptr<Dialog> dialog) **override**

This method implements the ui mode specific mechanism for showing a dialog.

Must be called in main thread.

## Private Slots

void **cmd\_\_change\_dialog\_property\_\_M** (*WebServerEngineRequest* \*request)

void **cmd\_\_closed\_in\_browser\_\_M** (*WebServerEngineRequest* \*request)

void **cmd\_\_list\_\_M** (*WebServerEngineRequest* \*request)

### 10.1.444 Class sh::ui::web::WebExceptionDialog

class *sh::ui::web::WebExceptionDialog* : public QObject, public *sh::ui::web::WebDialog*, public *sh::ui::Excepti*  
Web based exception dialog.

## Public Functions

**WebExceptionDialog** (QString *error1*, QString *error2*, QString *details*, QString *icon*, bool *mayRetry*, bool *mayClose*, bool *mayCancel*, bool *showLoglabelAndDetails*)

*ExceptionDialogResult* **answer** () **override**

Returns the answer the user has given in this dialog.

QVariant **dialogProperty** (QString *dlgPropertyKey*, QVariant *deflt* = QVariant())

Returns a dialog property value by key.

void **setDialogProperty** (QString *dlgPropertyKey*, QVariant *value*)

Sets a dialog property value for a key.

QVariantHash **dialogResult** ()

Returns the dialog result as hash.

The browser side of a web dialog can ‘accept’ a dialog while closing by setting a non-empty dialog result. This mechanism allows browser side dialogs to return data to the backend. It’s up to the backend to interpret the content of this hash.

If the dialog was cancelled, or is still open, the returned hash is empty. See also `wasClosed()`.

void **triggerDialogEvent** (QString *name*, QJsonObject *data* = QJsonObject())

Triggers a dialog event (at the browsers).

They in turn typically fetch some data again in order to refresh the ui.

void **setCloseableByUser** (bool *v* = true)

Sets if this dialog shall be directly closeable by the user by the window decoration. Set to `false` if the user has to react on this dialog by clicking on some controls in the dialog body.

This method must be called during construction.

bool **isCloseableByUser** ()

Returns if this dialog shall be directly closeable by the user by the window decoration.

QJsonObject **toJson** () **override**

Returns the json representation as QJsonObject.

QString **error1** ()

Returns the 1st error text.

QString **error2** ()

Returns the 2nd error text.

QString **details** ()

Returns the error details.

QString **icon** ()

Returns the icon (as name resolveable by *sh::base::IconManager*).

bool **mayRetry** ()

Returns if it’s allowed to retry.

bool **mayClose** ()

Returns if it’s allowed to just close and continue without closing Shallot.

bool **mayCancel** ()

Returns if it’s allowed to cancel, i.e. throwing a *sh::exceptions::CancelException*.

bool **showLoglabelAndDetails** ()

Returns if the dialog shall allow to read the details.

`qint64 dialogId ()`  
Returns the dialog id.  
Each instance has an id unique in the complete Shallot process lifetime.  
Must be called in main thread.

`bool isInitiated ()`  
Returns if this dialog is initialized.  
Must be called in main thread.

`bool wasAccepted ()`  
Returns if this dialog was 'accepted' by the user (typically by clicking some 'OK' button).  
Must be called in main thread.

`void waitClosed ()`  
Wait until the user closed the dialog in some way.  
May be called in any thread.

`void close ()`  
Closes the dialog.  
Must be called in main thread.

`bool wasClosed ()`  
Returns if this dialog was closed.  
Must be called in main thread.

*DialogManager* \*`manager ()`  
Returns the *DialogManager* which hosts this dialog.

### Private Slots

`void cmd__icon__M(WebServerEngineRequest *request)`

## 10.1.445 Class sh::ui::web::WebFilePropertyDialog

`class sh::ui::web::WebFilePropertyDialog : public QObject, public sh::ui::web::WebDialog, public sh::ui::FilePropertyDialog`  
Web based file property dialog.

### Public Functions

`void refresh () override`  
Refreshes the dialog content.

*FilePropertyDialogTabTableView* \*`createTabViewTable () override`  
Creates a new tab view table sub-widget.

*FilePropertyDialogTabTextView* \*`createTabViewText () override`  
Creates a new tab view text sub-widget.

*FilePropertyDialogTabIconTextBannerView* \*`createTabViewIconTextBanner () override`  
Creates a new tab view icon text banner sub-widget.

`WebFilePropertyDialog (QList<std::shared_ptr<sh::filesystem::FilesystemNode>> nodes)`

`~WebFilePropertyDialog ()`



QVariant **dialogProperty** (QString *dlgPropertyKey*, QVariant *deflt* = QVariant())

Returns a dialog property value by key.

void **setDialogProperty** (QString *dlgPropertyKey*, QVariant *value*)

Sets a dialog property value for a key.

QVariantHash **dialogResult** ()

Returns the dialog result as hash.

The browser side of a web dialog can ‘accept’ a dialog while closing by setting a non-empty dialog result. This mechanism allows browser side dialogs to return data to the backend. It’s up to the backend to interpret the content of this hash.

If the dialog was cancelled, or is still open, the returned hash is empty. See also `wasClosed()`.

void **triggerDialogEvent** (QString *name*, QJsonObject *data* = QJsonObject())

Triggers a dialog event (at the browsers).

They in turn typically fetch some data again in order to refresh the ui.

void **setCloseableByUser** (bool *v* = true)

Sets if this dialog shall be directly closeable by the user by the window decoration. Set to `false` if the user has to react on this dialog by clicking on some controls in the dialog body.

This method must be called during construction.

bool **isCloseableByUser** ()

Returns if this dialog shall be directly closeable by the user by the window decoration.

QJsonObject **toJson** () **override**

Returns the json representation as QJsonObject.

QList<std::shared\_ptr<FilePropertyDialogTab>> **tabs** ()

Returns a list of all tabs.

*sh::ui::FilePropertyDialogTabActionsView* \***widgetAt** (std::shared\_ptr<FilePropertyDialogTab> *tab*,  
int *i*)

Returns the widget from a tab at a certain index.

int **widgetCount** (std::shared\_ptr<FilePropertyDialogTab> *tab*)

Returns the number of widgets in a tab.

qint64 **dialogId** ()

Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

bool **isInited** ()

Returns if this dialog is initialized.

Must be called in main thread.

bool **wasAccepted** ()

Returns if this dialog was ‘accepted’ by the user (typically by clicking some ‘OK’ button).

Must be called in main thread.

void **waitClosed** ()

Wait until the user closed the dialog in some way.

May be called in any thread.

void **close** ()

Closes the dialog.

Must be called in main thread.

bool **wasClosed** ()

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \***manager** ()

Returns the *DialogManager* which hosts this dialog.

## Public Static Functions

void **addTabFactory** (int *i*, std::shared\_ptr<*FilePropertyDialogTabFactory*> *factory*)

Adds a *FilePropertyDialogTabFactory* so the Properties dialogs show its content.

See also the REGISTER\_FILEPROPERTYDIALOGTAB macro.

### Parameters

- *i*: An index which controls the display order. Use one of the REGISTER\_FILEPROPERTYDIALOGTAB\_INDEX\_\* values in *FilePropertyDialogTabFactory* as base values.

## Private Functions

void **tabwidgetTableSelect** (int *cookie*, int *tabidx*, int *widgetidx*, QList<int> *lsel*)

void **triggerAction** (int *cookie*, int *tabidx*, int *widgetidx*, int *btnidx*)

QJsonArray **getProperties** ()

## Private Members

int **\_cookie** = 0

QTimer **\_refreshTimer**

QList<*WebFilePropertyDialogChild*\*> **\_childs**

## Private Slots

void **cmd\_\_get\_properties\_\_M** (*WebServerEngineRequest* \**request*)

void **cmd\_\_refresh\_\_M** (*WebServerEngineRequest* \**request*)

void **cmd\_\_trigger\_action\_\_M** (*WebServerEngineRequest* \**request*)

void **cmd\_\_tabwidget\_table\_select\_\_M** (*WebServerEngineRequest* \**request*)

**class WebFilePropertyDialogChild**

Subclassed by *sh::ui::web::WebFilePropertyDialog::WebFilePropertyDialogTabActionsView*,  
*sh::ui::web::WebFilePropertyDialog::WebFilePropertyDialogTabIconTextBannerView*,  
*sh::ui::web::WebFilePropertyDialog::WebFilePropertyDialogTabTableView*,  
*sh::ui::web::WebFilePropertyDialog::WebFilePropertyDialogTabTextView*

## Public Functions

**WebFilePropertyDialogChild** (*WebFilePropertyDialog* \*dlg)

**~WebFilePropertyDialogChild** ()

**class** **WebFilePropertyDialogTabActionsView**: **public** *sh::ui::FilePropertyDialogTabActionsView*, **public** *sh::ui::FilePropertyDialogTabActionsView*

## Public Functions

**WebFilePropertyDialogTabActionsView** (*WebFilePropertyDialog* \*dlg)

**void** **setVisible** (bool v) **override**  
Sets the view visible or hidden. .

**void** **setContent** (*FilePropertyDialogTabViewContent* \*cnt) **override**  
Sets the main widget. .

**void** **setButtons** (QList<QString> buttons) **override**  
Sets the list of buttons. .

**void** **setLabelText** (QString text)

**QJsonObject** **toJson** () **override**  
Returns the json representation as QJsonObject.

*FilePropertyDialogTabViewContent* \***content** ()  
The main widget.

QList<QString> **buttons** ()  
The list of buttons.

**void** **onButtonTriggered** (std::function<void> int i  
> fct, QObject \*owner = 0) Sets a handler for a click on one of the buttons (optionally bound to an owner lifetime).

## Private Members

bool **\_visible** = false

QString **\_labelText**

## Friends

**friend class** WebFilePropertyDialog

**class** **WebFilePropertyDialogTabIconTextBannerView**: **public** *sh::ui::FilePropertyDialogTabIconTextBannerView*

### Public Functions

**WebFilePropertyDialogTabIconTextBannerView** (*WebFilePropertyDialog* \*dlg)

void **clear** () **override**  
Clears the contents.

void **insertIcon** (QIcon *icon*, int *sizePt* = 24) **override**  
Adds an icon.

void **insertText** (QString *text*) **override**  
Adds a text.

QJsonObject **toJson** () **override**  
Returns the json representation as QJsonObject.

### Private Members

QList<*WebIconTextBannerItem*> **\_content**

**class WebFilePropertyDialogTabTableView** : public *sh::ui::FilePropertyDialogTabTableView*, public *sh::tools::Tool*

### Public Types

**typedef** QPair<int, int> **ItemIndex**

### Public Functions

**WebFilePropertyDialogTabTableView** (*WebFilePropertyDialog* \*dlg)

void **setContent** (QList<QPair<QString, QString>> *content*) **override**  
Sets the content.

QList<*ItemIndex*> **getSelectedItems** () **override**  
Returns the selected cells.

QVariant **getItemValue** (int *r*, int *c*) **override**  
Returns a value of a cell.

QJsonObject **toJson** () **override**  
Returns the json representation as QJsonObject.

void **setSelectedItems** (QList<int> *selItems*)

### Private Members

QList<QPair<QString, QString>> **\_content**

QList<int> **\_selectedItems**

**class WebFilePropertyDialogTabTextView** : public *sh::ui::FilePropertyDialogTabTextView*, public *sh::tools::Tool*

## Public Functions

**WebFilePropertyDialogTabTextView** (*WebFilePropertyDialog \*dlg*)

void **setContent** (QString *content*) **override**  
Sets the content.

QJsonObject **toJson** () **override**  
Returns the json representation as QJsonObject.

## Private Members

QString **\_content**

### 10.1.446 Class sh::ui::web::WebFilePropertyDialog::WebFilePropertyDialogChild

**class** *sh::ui::web::WebFilePropertyDialog::WebFilePropertyDialogChild*

Subclassed by *sh::ui::web::WebFilePropertyDialog::WebFilePropertyDialogTabActionsView*,  
*sh::ui::web::WebFilePropertyDialog::WebFilePropertyDialogTabIconTextBannerView*,  
*sh::ui::web::WebFilePropertyDialog::WebFilePropertyDialogTabTableView*, *sh::ui::web::WebFilePropertyDialog::WebFileProp*

## Public Functions

**WebFilePropertyDialogChild** (*WebFilePropertyDialog \*dlg*)

**~WebFilePropertyDialogChild** ()

### 10.1.447 Class sh::ui::web::WebFilePropertyDialog::WebFilePropertyDialogTabActionsView

**class** *sh::ui::web::WebFilePropertyDialog::WebFilePropertyDialogTabActionsView* : public *sh::ui::F*

## Public Functions

**WebFilePropertyDialogTabActionsView** (*WebFilePropertyDialog \*dlg*)

void **setVisible** (bool *v*) **override**  
Sets the view visible or hidden. .

void **setContent** (*FilePropertyDialogTabViewContent \*cnt*) **override**  
Sets the main widget. .

void **setButtons** (QList<QString> *buttons*) **override**  
Sets the list of buttons. .

void **setLabelText** (QString *text*)

QJsonObject **toJson** () **override**  
Returns the json representation as QJsonObject.

*FilePropertyDialogTabViewContent \*content* ()  
The main widget.

QList<QString> **buttons** ()  
The list of buttons.

```
void onButtonTriggered (std::function<void>) int i  
    > fcn, QObject *owner = 0 Sets a handler for a click on one of the buttons (optionally bound to an owner  
    lifetime).
```

### Private Members

```
bool _visible = false  
QString _labelText
```

### Friends

```
friend class WebFilePropertyDialog
```

## 10.1.448 Class sh::ui::web::WebFilePropertyDialog::WebFilePropertyDialogTabIconTextBannerView

```
class sh::ui::web::WebFilePropertyDialog::WebFilePropertyDialogTabIconTextBannerView : public
```

### Public Functions

```
WebFilePropertyDialogTabIconTextBannerView (WebFilePropertyDialog *dlg)
```

```
void clear () override  
    Clears the contents.
```

```
void insertIcon (QIcon icon, int sizePt = 24) override  
    Adds an icon.
```

```
void insertText (QString text) override  
    Adds a text.
```

```
QJsonObject toJson () override  
    Returns the json representation as QJsonObject.
```

### Private Members

```
QList<WebIconTextBannerItem> _content
```

## 10.1.449 Class sh::ui::web::WebFilePropertyDialog::WebFilePropertyDialogTabTableView

```
class sh::ui::web::WebFilePropertyDialog::WebFilePropertyDialogTabTableView : public sh::ui::File
```

## Public Types

```
typedef QPair<int, int> ItemIndex
```

## Public Functions

```
WebFilePropertyDialogTabTableView (WebFilePropertyDialog *dlg)
```

```
void setContent (QList<QPair<QString, QString>> content) override  
    Sets the content.
```

```
QList<ItemIndex> getSelectedItems () override  
    Returns the selected cells.
```

```
QVariant getItemValue (int r, int c) override  
    Returns a value of a cell.
```

```
QJsonObject toJson () override  
    Returns the json representation as QJsonObject.
```

```
void setSelectedItems (QList<int> selitms)
```

## Private Members

```
QList<QPair<QString, QString>> _content
```

```
QList<int> _selectedItems
```

### 10.1.450 Class sh::ui::web::WebFilePropertyDialog::WebFilePropertyDialogTabTextView

```
class sh::ui::web::WebFilePropertyDialog::WebFilePropertyDialogTabTextView : public sh::ui::FileP
```

## Public Functions

```
WebFilePropertyDialogTabTextView (WebFilePropertyDialog *dlg)
```

```
void setContent (QString content) override  
    Sets the content.
```

```
QJsonObject toJson () override  
    Returns the json representation as QJsonObject.
```

## Private Members

```
QString _content
```

### 10.1.451 Class `sh::ui::web::WebFileView`

**class** `sh::ui::web::WebFileView` : **public** `sh::ui::FileView`  
Web based file view.

#### Public Functions

**WebFileView** (`WebFileViewManager` \**manager*)

`sh::ui::ColumnDimensions` \***columnDimensions** () **override**  
Returns the column dimensions handler. .

`sh::tools::HistoryTracker`<std::shared\_ptr<const `sh::filesystem::Eurl`>> \***historyTracker** () **override**  
Returns the history tracker. .

`FileViewMode` **viewmode** () **override**  
Returns the view mode (icons, list, ... ?). .

void **setViewmode** (`FileViewMode` *m*) **override**  
Sets the view mode. .

int **index** () **override**  
Returns the position of this file view within the panel. .

void **gotoDir** (std::shared\_ptr<`sh::filesystem::FilesystemNode`> *n*) **override**  
Jumps to a new current directory.  
  
Implementations have to call the base class implementation inside.

`sh::filesystem::SizeFormatting` **getSizeFormattingMode** () **override**  
Returns the mode how file sizes are formatted for displaying. .

void **setSizeFormattingMode** (`sh::filesystem::SizeFormatting` *mode*) **override**  
Sets the mode how file sizes are formatted for displaying. .

bool **hiddenFilesVisible** () **override**  
Returns if hidden files are visible. .

void **setHiddenFilesVisible** (bool *v*) **override**  
Sets the visibility of hidden files. .

int **iconDimension** () **override**  
Returns the icon size (in pixels). .

void **setIconDimension** (int *v*) **override**  
Sets the icon size (in pixels). .

void **setSort** (int *column*, Qt::SortOrder *order*) **override**  
Sets how to sort this view. .

int **sortColumn** () **override**  
Returns the index of the current sort column. .

Qt::SortOrder **sortOrder** () **override**  
Returns the current sort order. .

QList<std::shared\_ptr<`sh::filesystem::FilesystemNode`>> **selectedNodes** () **override**  
Returns the nodes which the user has selected in this fileview. .

void **setSelection** (const QList<std::shared\_ptr<`sh::filesystem::FilesystemNode`>> *nodes*) **override**  
Sets the node selection of this fileview. .



`QList<std::shared_ptr<sh::filesystem::FilesystemNode>> getAllVisibleNodes () override`  
Returns a list of all nodes currently listed in this file view. .

`qint64 id ()`  
Returns the unique identifier.

`std::shared_ptr<sh::filesystem::FilesystemModelFileviewProxy> model ()`  
Returns the filesystem model bound to this view.

`std::shared_ptr<sh::filesystem::FilesystemNode> node ()`  
Returns the current directory.

`void reload (bool skipModel = false)`  
Reloads the data inside this file view.

`sh::filesystem::FilesystemModelFileviewProxy *filemodel ()`  
Returns the filesystem model for this view. .

`void setFilemodel (sh::filesystem::FilesystemModelFileviewProxy *model)`  
Sets the filesystem model for this view. .

## Signals

`void currentNodeChanged ()`

`void modelChanged ()`

`void selectionChanged ()`  
Triggered when the selection have changed.

`void viewOptionChanged ()`  
Triggered when some view options have changed.

## Private Functions

`void __set_model ()`

`void setSelectedNode (std::shared_ptr<sh::filesystem::FilesystemNode> node)`

`void setCheckedNodes (QList<std::shared_ptr<sh::filesystem::FilesystemNode>> nodes)`

## Private Members

`std::shared_ptr<sh::filesystem::FilesystemModelFileviewProxy> _model`

`qint64 _id`

`FileViewMode _viewmode = FileViewMode::ListView`

`sh::filesystem::SizeFormatting _sizeformatting = sh::filesystem::SizeFormatting::SizeFormattingModePrefixes`

`bool _hiddenFilesVisible = false`

`int _iconDimension = 20`

`int _sortColumn = 0`

`Qt::SortOrder _sortOrder = Qt::SortOrder::AscendingOrder`

`std::shared_ptr<sh::filesystem::FilesystemNode> _selectedNode`

`QList<std::shared_ptr<sh::filesystem::FilesystemNode>> _checkedNodes`

*WebFileViewManager* \*\_manager

## Friends

**friend class** WebFileViewManager

### 10.1.452 Class sh::ui::web::WebFileViewManager

**class** *sh::ui::web::WebFileViewManager* : public QObject, public *sh::ui::web::WebModule*  
Manages *WebFileView* instances.

## Public Functions

**bool** **executeCommand** (*WebServerEngineRequest* \*request, QString command) **override**  
Executes command as requested by the browser, if responsible.

Read parameters, check if this module is responsible for answering, and then write an answer with request.

Implementations often use *executeCommand\_byQObjectReflection()* for convenience.

**Return** If it handled (answered) the command.

std::shared\_ptr<*WebFileView*> **addFileView** (bool *reinitialize*)

void **setCurrentFileViewByIndex** (int *idx*)

*sh::ui::web::WebFileView* \***currentFileView** ()

int **fileViewCount** ()

void **removeFileView** (int *i*)

*sh::ui::web::WebFileView* \***getFileView** (int *i*)

int **getFileViewIndex** (*sh::ui::FileView* \*fileview)

**bool** **rootCommand** (*WebServerEngineRequest* \*request, QString webadapter)  
Returns the root ('index.html') content. .

**void** **setEngine** (*WebServerEngine* \*webserver)  
Assigns this web module to a *sh::ui::web::WebServerEngine*. .

*WebServerEngine* \***webserver** ()  
Returns the *sh::ui::web::WebServerEngine* this web module is assigned to (may be nullptr).

## Signals

void **activeSelectionChanged** ()

void **fileViewOptionsChanged** (int *i*)

void **fileViewCountChanged** ()

void **currentFileViewChanged** ()

## Public Static Functions

bool **executeCommand\_byQObjectReflection** (QObject \*o, *WebServerEngineRequest* \*request, QString command, QString justLeftside = QString())

Convenience function often used by *executeCommand()*.

For a QObject, it searches there for a slot with the name cmd\_{command}, with `/'s replaced by '\_'s (so, for the command "foo/get\_bar", it searches for slotcmd\_foo\_get\_bar). If it finds one, it gets executed (in a worker thread). The request is considered as handled, if so.

If there is such a slot, but with appended \_\_M in name, it is called in main thread!

**Return** If it handled (answered) the command.

## Private Members

QList<std::shared\_ptr<*WebFileView*>> **\_fileviews**

int **\_icurrentfileview** = 0

## Private Slots

void **cmd\_\_checkednodes\_changed\_\_M**(*WebServerEngineRequest* \*request)

void **cmd\_\_focus\_current\_\_M**(*WebServerEngineRequest* \*request)

void **cmd\_\_get\_\_M**(*WebServerEngineRequest* \*request)

void **cmd\_\_list\_view\_items\_\_M**(*WebServerEngineRequest* \*request)

void **cmd\_\_node\_activated\_\_M**(*WebServerEngineRequest* \*request)

void **cmd\_\_selection\_changed\_\_M**(*WebServerEngineRequest* \*request)

## 10.1.453 Class sh::ui::web::WebI18n

**class** *sh::ui::web::WebI18n*

Web related helpers for internationalization/translations.

## Public Static Functions

QString **get** ()

Returns the Javascript representation for translated strings (in current ui language).

## 10.1.454 Class `sh::ui::web::WebLogViewDialog`

**class** `sh::ui::web::WebLogViewDialog` : **public** `QObject`, **public** `sh::ui::web::WebDialog`, **public** `sh::ui::LogViewDialog`  
Web based log view dialog.

### Public Functions

**WebLogViewDialog** (`QString headtext`, `QString subheadtxt`, `sh::base::LogSeverity minseverity`)

`QVariant dialogProperty` (`QString dlgPropertyKey`, `QVariant deflt = QVariant()`)  
Returns a dialog property value by key.

`void setDialogProperty` (`QString dlgPropertyKey`, `QVariant value`)  
Sets a dialog property value for a key.

`QVariantHash dialogResult` ()  
Returns the dialog result as hash.

The browser side of a web dialog can ‘accept’ a dialog while closing by setting a non-empty dialog result. This mechanism allows browser side dialogs to return data to the backend. It’s up to the backend to interpret the content of this hash.

If the dialog was cancelled, or is still open, the returned hash is empty. See also `wasClosed()`.

`void triggerDialogEvent` (`QString name`, `QJsonObject data = QJsonObject()`)  
Triggers a dialog event (at the browsers).

They in turn typically fetch some data again in order to refresh the ui.

`void setCloseableByUser` (`bool v = true`)  
Sets if this dialog shall be directly closeable by the user by the window decoration. Set to `false` if the user has to react on this dialog by clicking on some controls in the dialog body.

This method must be called during construction.

`bool isCloseableByUser` ()  
Returns if this dialog shall be directly closeable by the user by the window decoration.

`QJsonObject toJson` () **override**  
Returns the json representation as `QJsonObject`.

`QString headtext` ()  
Returns the header text.

`QString subheadtxt` ()  
Returns the 2nd header text.

`sh::base::LogSeverity minimumSeverity` ()  
Returns the minimum severity.

`qint64 dialogId` ()  
Returns the dialog id.  
Each instance has an id unique in the complete Shallot process lifetime.  
Must be called in main thread.

`bool isInitd` ()  
Returns if this dialog is initialized.  
Must be called in main thread.

bool **wasAccepted** ()  
Returns if this dialog was ‘accepted’ by the user (typically by clicking some ‘OK’ button).  
Must be called in main thread.

void **waitClosed** ()  
Wait until the user closed the dialog in some way.  
May be called in any thread.

void **close** ()  
Closes the dialog.  
Must be called in main thread.

bool **wasClosed** ()  
Returns if this dialog was closed.  
Must be called in main thread.

*DialogManager* \***manager** ()  
Returns the *DialogManager* which hosts this dialog.

### Private Slots

void **cmd\_\_get\_messages** (*WebServerEngineRequest* \*request)

## 10.1.455 Class sh::ui::web::WebMainWindow

**class** *sh::ui::web::WebMainWindow* : **public** QObject, **public** *sh::ui::MainWindow*, **public** *sh::ui::web::WebModule*  
The web main window implementation.

### Public Functions

**WebMainWindow** (QString *dispatcherUrl*, QByteArray *wtoken*)

**~WebMainWindow** ()

*WebServerEngine* \***webserver** ()  
Returns the *WebServerEngine* instance for this main window.

void **initialize** () **override**  
Initializes this main window. .

std::shared\_ptr<*DialogManager*> **dialogManager** () **override**  
Returns the *DialogManager* which creates and drives *Dialogs* for this main window.  
You typically should not need to use it directly.

int **fileViewCount** () **override**  
Returns the current number of file views. .

void **addFileView** (bool *reinitialize* = false) **override**  
Adds a new file view. .

void **removeFileView** (int *i*) **override**  
Removes a file view. .

*sh::ui::FileView* \***currentFileView** () **override**  
Returns the file view which is currently active (i.e. focussed). .

*sh::ui::FileView* \***getFileView** (int *i*) **override**  
Returns a file view by position index. .

void **jumpToNode** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> *node*) **override**  
Let the current view jump to another location. .

std::shared\_ptr<*sh::filesystem::FilesystemNode*> **currentDirectory** () **override**  
Gets the *sh::filesystem::FilesystemNode* selected in the current view. .

void **setTitlePattern** (QString *value*) **override**  
Sets the window title pattern. .

void **setTreeVisible** (bool *v*) **override**  
Sets the visibility of the directory tree. .

void **setTreeSticky** (bool *v*) **override**  
Sets if the directory tree should follow the active file view. .

void **setFileDetailsPanelVisible** (bool *v*) **override**  
Sets the visibility of the file details panel. .

std::shared\_ptr<*sh::ui::ActionExecutionInfoPanel*> **addInfoPanel** (int *position*,  
*sh::actions::ActionExecutionInfo*  
\**info*, QColor *color* = QColor())  
**override**  
Creates a new action execution panel.  
Let this smart pointer die for removing it.

void **jumpbarSetTextMode** () **override**  
Sets the jumpbar to text input mode. .

void **jumpbarSetButtonMode** () **override**  
Sets the jumpbar to button mode. .

bool **jumpbarIsTextMode** () **override**  
Returns if the jumpbar is in text input mode. .

QString **toolbarPosition** () **override**  
Returns the current toolbar position. .

void **setToolbarPosition** (QString *pos*) **override**  
Sets the toolbar position. .

QString **detailPanelPosition** () **override**  
Returns the current detail panel position. .

void **setDetailPanelPosition** (QString *pos*) **override**  
Sets the detail panel position. .

std::shared\_ptr<*AboutDialog*> **showAboutDialog** () **override**  
Shows and returns an ‘About’ dialog. .

std::shared\_ptr<*ManageProfilesDialog*> **showManageProfilesDialog** () **override**  
Shows and returns a ‘Manage Saved Settings’ dialog. .

std::shared\_ptr<*OpenWithDialog*> **showOpenWithDialog** (std::shared\_ptr<*sh::filesystem::FilesystemNode*>  
*node*, QList<std::shared\_ptr<*sh::tools::filetypes::OpenMethod*>>  
*openMethods*) **override**  
Shows and returns a ‘Open With’ dialog. .

```

std::shared_ptr<StoreProfileDialog> showStoreProfileDialog (std::shared_ptr<const
                                                    sh::filesystem::Eurl> eurl)
                                                    override

    Shows and returns a 'Save Settings' dialog. .

std::shared_ptr<TuningDialog> showTuningDialog () override
    Shows and returns a 'Tuning' dialog. .

std::shared_ptr<LogViewDialog> showLogViewDialog (QString headtext = QString(),
                                                    QString subheadtxt = QString(),
                                                    sh::base::LogSeverity minseverity =
                                                    sh::base::LogSeverity::_DEFAULT)
                                                    override

    Shows and returns a 'Log' dialog. .

std::shared_ptr<ManageBookmarksDialog> showManageBookmarksDialog () override
    Shows and returns a 'Manage Bookmarks' dialog. .

std::shared_ptr<FilePropertyDialog> showFilePropertyDialog (QList<std::shared_ptr<sh::filesystem::FilesystemNode>>
                                                    nodes) override

    Shows and returns a 'File Properties' dialog. .

std::shared_ptr<ActionExecutionInfoDialog> createActionExecutionInfoDialog (sh::actions::ActionExecutionInfo
                                                    *info)
                                                    override

    Creates an action execution dialog. .

void handleCloseAppRejected (QString rejectmsg) override
    React on a rejected application close request (with some feedback message). .

void _initialize (sh::base::SingletonInitializer *singletonInitializer)
    Initializes the main window. For custom initialization logic, see MainWindow.initialize. .

void fileViewsReloadAll ()
    Reload all content in each file view.

void onFileViewOptionsChanged (std::function<void> int
    > fct, QObject *owner = 0) Sets a handler for some view options in a file view changed (optionally bound
    to an owner lifetime).

void onCurrentFileViewChanged (std::function<void>
    > fct, QObject *owner = 0) Sets a handler for the currently active file view changed (optionally bound to an
    owner lifetime).

void onFileViewCountChanged (std::function<void>
    > fct, QObject *owner = 0) Sets a handler for the number of file views changed (optionally bound to an owner
    lifetime).

void onCurrentDirectoryChanged (std::function<void>
    > fct, QObject *owner = 0) Sets a handler for the current directory in the active file view changed (optionally
    bound to an owner lifetime).

void onGlobalViewOptionsChanged (std::function<void>
    > fct, QObject *owner = 0) Sets a handler for some global view options changed (optionally bound to an
    owner lifetime).

void onCurrentProfileChanged (std::function<void>
    > fct, QObject *owner = 0) Sets a handler for the current profile changed (optionally bound to an owner
    lifetime).

void jumpToEurl (std::shared_ptr<const sh::filesystem::Eurl> eurl)
    Let the current view jump to another location. .

```

`QString titlePattern ()`  
Returns the window title pattern.

`void setCurrentProfile (QString profile)`  
Sets the current profile. .

`QString currentProfile ()`  
Returns the current profile.

`void reloadProfile ()`  
Reloads the profile data.

`bool treeVisible ()`  
Returns the visibility of the directory tree.

`bool treeSticky ()`  
Returns if the directory tree follows the active file view.

`bool fileDetailsPanelVisible ()`  
Returns the visibility of the file details panel.

`std::shared_ptr<sh::ui::ActionExecutionInfoPanel> addInfoPanel (sh::actions::ActionExecutionInfo *info, QColor color = QColor())`  
Creates a new action execution panel.  
Let this smart pointer die for removing it.

`std::shared_ptr<ActionExecutionInfoPanel> showErrorPanel ()`  
Shows an error panel.

`void _closeDirectly ()`  
Directly begin application shutdown without checks.

`bool closeApp ()`  
Check if the application may shut down now, and if so, initiate it.

`bool isCloseable (QString *msg = nullptr)`  
Check if the application may shut down now.

`void handleClosed ()`  
Do some cleanup steps just when closing starts. Implementations must call base implementation!

`bool isClosing ()`  
Returns if the application is currently closing.

`bool rootCommand (WebServerEngineRequest *request, QString webadapter)`  
Returns the root ('index.html') content. .

`void setEngine (WebServerEngine *webserver)`  
Assigns this web module to a *sh::ui::web::WebServerEngine*. .

## Public Static Functions

`bool tryCreateMainWindow (MainWindow *&mainWindow)`  
*WebMainWindow* \*mainWindow ()

`void openUrlOnDesktop (QUrl url)`  
Opens a url on the user desktop, typically in the browser.  
Note: It obeys the 'openbrowser' ui parameter.



void **setMainWindow** (*MainWindow \*mainWindow*)  
Sets the main window instance. .

bool **isReady** ()  
Returns if this process ui is ready (i.e. is created or runs headless).

bool **runsHeadless** ()  
Returns if this process runs headless, i.e. without any ui, just for a background process.

void **\_closeDirectly** (*sh::base::SingletonInitializer \*singletonInitializer*)

QString **uiMode** ()  
Returns the UI mode (e.g. qt, web).

bool **executeCommand\_byQObjectReflection** (QObject \*o, *WebServerEngineRequest \*request*, QString *command*, QString *justLeftside* = QString())  
Convenience function often used by *executeCommand()*.  
For a QObject, it searches there for a slot with the name cmd\_{command}, with ``'s replaced by '\_'s (so, for the command "foo/getBars", it searches for slotcmd\_foo\_getBars). If it finds one, it gets executed (in a worker thread). The request is considered as handled, if so.  
If there is such a slot, but with appended \_\_M in name, it is called in main thread!  
**Return** If it handled (answered) the command.

## Private Functions

std::shared\_ptr<*sh::ui::web::WebActionManager*> **actionManager** ()

void **setCloseable** (bool *closeable*, QString *message* = QString()) **override**  
Sets the closeable state and message.

void **setCurrentEurlByNode** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> *node*)

## Private Members

bool **\_treeSticky** = true

bool **\_detailsPanelVisible** = true

QString **\_currentProfile**

*WebServerEngine* \* **\_webserver**

std::shared\_ptr<*sh::filesystem::FilesystemNode*> **\_currentDirectory** = 0

std::shared\_ptr<*WebActionManager*> **\_actionManager**

std::shared\_ptr<*WebDetailPanelManager*> **\_detailPanelManager**

std::shared\_ptr<*WebDialogManager*> **\_dialogManager**

std::shared\_ptr<*WebFileViewManager*> **\_fileViewManager**

QString **\_lastCloseableState**

### Private Slots

```
void cmd_filesystem_get_icon (WebServerEngineRequest *request)
void cmd_filesystem_list_items (WebServerEngineRequest *request)
void cmd_log_as_text (WebServerEngineRequest *request)
void cmd_log_log_message (WebServerEngineRequest *request)
void cmd_mainwindow_set_current_directory (WebServerEngineRequest *request)
void cmd_mainwindow_shallot_icon (WebServerEngineRequest *request)
void cmd_mainwindow_show_logview (WebServerEngineRequest *request)
```

### Private Static Attributes

```
WebMainWindow *_webMainWindow = nullptr
```

### Friends

```
friend class WebFileViewManager
friend class WebActionExecutionInfoPanel
friend class WebActionExecutionInfoDialog
friend class WebDetailPanelManager
```

## 10.1.456 Class sh::ui::web::WebManageBookmarksDialog

```
class sh::ui::web::WebManageBookmarksDialog : public QObject, public sh::ui::web::WebDialog, public sh::ui::web::WebDialogPrivate
    Web based manage bookmarks dialog.
```

### Public Functions

```
WebManageBookmarksDialog ()
```

```
QVariant dialogProperty (QString dlgPropertyKey, QVariant deflt = QVariant())
    Returns a dialog property value by key.
```

```
void setDialogProperty (QString dlgPropertyKey, QVariant value)
    Sets a dialog property value for a key.
```

```
QVariantHash dialogResult ()
    Returns the dialog result as hash.
```

The browser side of a web dialog can ‘accept’ a dialog while closing by setting a non-empty dialog result. This mechanism allows browser side dialogs to return data to the backend. It’s up to the backend to interpret the content of this hash.

If the dialog was cancelled, or is still open, the returned hash is empty. See also wasClosed().

```
void triggerDialogEvent (QString name, QJsonObject data = QJsonObject())
    Triggers a dialog event (at the browsers).
```

They in turn typically fetch some data again in order to refresh the ui.

void **setCloseableByUser** (bool v = true)  
 Sets if this dialog shall be directly closeable by the user by the window decoration. Set to `false` if the user has to react on this dialog by clicking on some controls in the dialog body.

This method must be called during construction.

bool **isCloseableByUser** ()  
 Returns if this dialog shall be directly closeable by the user by the window decoration.

QJsonObject **toJson** () **override**  
 Returns the json representation as QJsonObject.

qint64 **dialogId** ()  
 Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

bool **isInited** ()  
 Returns if this dialog is initialized.

Must be called in main thread.

bool **wasAccepted** ()  
 Returns if this dialog was 'accepted' by the user (typically by clicking some 'OK' button).

Must be called in main thread.

void **waitClosed** ()  
 Wait until the user closed the dialog in some way.

May be called in any thread.

void **close** ()  
 Closes the dialog.

Must be called in main thread.

bool **wasClosed** ()  
 Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \***manager** ()  
 Returns the *DialogManager* which hosts this dialog.

### Private Slots

void **cmd\_\_add\_bookmark\_\_M** (*WebServerEngineRequest* \*request)  
 void **cmd\_\_delete\_bookmark** (*WebServerEngineRequest* \*request)  
 void **cmd\_\_get** (*WebServerEngineRequest* \*request)  
 void **cmd\_\_move\_bookmark\_down** (*WebServerEngineRequest* \*request)  
 void **cmd\_\_move\_bookmark\_to\_folder** (*WebServerEngineRequest* \*request)  
 void **cmd\_\_move\_bookmark\_up** (*WebServerEngineRequest* \*request)  
 void **cmd\_\_store** (*WebServerEngineRequest* \*request)

### 10.1.457 Class `sh::ui::web::WebManageProfilesDialog`

**class** `sh::ui::web::WebManageProfilesDialog` : **public** `QObject`, **public** `sh::ui::web::WebDialog`, **public** `sh::ui::`  
Web based manage saved settings dialog.

#### Public Functions

**WebManageProfilesDialog** ()

`QVariant dialogProperty` (`QString dlgPropertyKey`, `QVariant deflt = QVariant()`)  
Returns a dialog property value by key.

**void setDialogProperty** (`QString dlgPropertyKey`, `QVariant value`)  
Sets a dialog property value for a key.

`QVariantHash dialogResult` ()  
Returns the dialog result as hash.

The browser side of a web dialog can ‘accept’ a dialog while closing by setting a non-empty dialog result. This mechanism allows browser side dialogs to return data to the backend. It’s up to the backend to interpret the content of this hash.

If the dialog was cancelled, or is still open, the returned hash is empty. See also `wasClosed()`.

**void triggerDialogEvent** (`QString name`, `QJsonObject data = QJsonObject()`)  
Triggers a dialog event (at the browsers).

They in turn typically fetch some data again in order to refresh the ui.

**void setCloseableByUser** (`bool v = true`)  
Sets if this dialog shall be directly closeable by the user by the window decoration. Set to `false` if the user has to react on this dialog by clicking on some controls in the dialog body.

This method must be called during construction.

**bool isCloseableByUser** ()  
Returns if this dialog shall be directly closeable by the user by the window decoration.

`QJsonObject toJson` () **override**  
Returns the json representation as `QJsonObject`.

`qint64 dialogId` ()  
Returns the dialog id.  
  
Each instance has an id unique in the complete Shallot process lifetime.  
  
Must be called in main thread.

**bool isInited** ()  
Returns if this dialog is initialized.  
  
Must be called in main thread.

**bool wasAccepted** ()  
Returns if this dialog was ‘accepted’ by the user (typically by clicking some ‘OK’ button).  
  
Must be called in main thread.

**void waitClosed** ()  
Wait until the user closed the dialog in some way.  
  
May be called in any thread.

void **close** ()  
 Closes the dialog.  
 Must be called in main thread.

bool **wasClosed** ()  
 Returns if this dialog was closed.  
 Must be called in main thread.

*DialogManager* \***manager** ()  
 Returns the *DialogManager* which hosts this dialog.

### Private Slots

void **cmd\_\_get\_\_M** (*WebServerEngineRequest* \*request)  
 void **cmd\_\_icon** (*WebServerEngineRequest* \*request)  
 void **cmd\_\_remove\_node\_\_M** (*WebServerEngineRequest* \*request)  
 void **cmd\_\_remove\_profile\_\_M** (*WebServerEngineRequest* \*request)

## 10.1.458 Class sh::ui::web::WebModule

**class** *sh::ui::web::WebModule*

Base class for modules, which provide some functionality on top of a web server engine.

Subclassed by *sh::ui::web::WebActionManager*, *sh::ui::web::WebDetailPanelManager*,  
*sh::ui::web::WebDialogManager*, *sh::ui::web::WebFileViewManager*, *sh::ui::web::WebMainWindow*,  
*sh::ui::web::WebServerEngineDispatcher*, *sh::ui::web::WebServerEngineMainModule*

### Public Functions

**WebModule** ()  
**~WebModule** ()

bool **executeCommand** (*WebServerEngineRequest* \*request, QString command)  
 Executes command as requested by the browser, if responsible.  
 Read parameters, check if this module is responsible for answering, and then write an answer with request.  
 Implementations often use *executeCommand\_byQObjectReflection()* for convenience.  
**Return** If it handled (answered) the command.

bool **rootCommand** (*WebServerEngineRequest* \*request, QString webadapter)  
 Returns the root ('index.html') content. .

void **setEngine** (*WebServerEngine* \*webserver)  
 Assigns this web module to a *sh::ui::web::WebServerEngine*. .

*WebServerEngine* \***webserver** ()  
 Returns the *sh::ui::web::WebServerEngine* this web module is assigned to (may be nullptr).

## Public Static Functions

bool **executeCommand\_byQObjectReflection** (QObject \*o, *WebServerEngineRequest* \*request, QString command, QString justLeftside = QString())

Convenience function often used by *executeCommand()*.

For a QObject, it searches there for a slot with the name cmd\_{command}, with `/'s replaced by '\_'s (so, for the command "foo/get\_bars", it searches for slotcmd\_foo\_get\_bars). If it finds one, it gets executed (in a worker thread). The request is considered as handled, if so.

If there is such a slot, but with appended \_\_M in name, it is called in main thread!

**Return** If it handled (answered) the command.

## Private Members

*WebServerEngine* \*\_webserver = nullptr

### 10.1.459 Class sh::ui::web::WebOpenWithDialog

class *sh::ui::web::WebOpenWithDialog* : public QObject, public *sh::ui::web::WebDialog*, public *sh::ui::OpenWithDialog*  
Web based open with dialog.

## Public Functions

**WebOpenWithDialog** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> node,  
QList<std::shared\_ptr<*sh::tools::filetypes::OpenMethod*>> openMethods)

std::shared\_ptr<*sh::tools::filetypes::OpenMethod*> **chosenMethod** () **override**  
Returns the chosen open-method (program for opening the file).

bool **rememberForMimetype** () **override**  
Returns if the chosen method is checked to be remembered for the same mime-type in the future.

std::shared\_ptr<const *sh::filesystem::Eurl*> **rememberForDirectory** () **override**  
If it's checked for the chosen method to be remembered for some subdirectory in the future, it returns the Eurl of this subdirectory, otherwise nullptr.

bool **rememberForFile** () **override**  
Returns if the chosen method is checked to be remembered for the same file in the future.

QVariant **dialogProperty** (QString dlgPropertyKey, QVariant deflt = QVariant())  
Returns a dialog property value by key.

void **setDialogProperty** (QString dlgPropertyKey, QVariant value)  
Sets a dialog property value for a key.

QVariantHash **dialogResult** ()  
Returns the dialog result as hash.

The browser side of a web dialog can 'accept' a dialog while closing by setting a non-empty dialog result. This mechanism allows browser side dialogs to return data to the backend. It's up to the backend to interpret the content of this hash.

If the dialog was cancelled, or is still open, the returned hash is empty. See also wasClosed().

void **triggerDialogEvent** (QString *name*, QJsonObject *data* = QJsonObject())

Triggers a dialog event (at the browsers).

They in turn typically fetch some data again in order to refresh the ui.

void **setCloseableByUser** (bool *v* = true)

Sets if this dialog shall be directly closeable by the user by the window decoration. Set to `false` if the user has to react on this dialog by clicking on some controls in the dialog body.

This method must be called during construction.

bool **isCloseableByUser** ()

Returns if this dialog shall be directly closeable by the user by the window decoration.

QJsonObject **toJson** () **override**

Returns the json representation as QJsonObject.

std::shared\_ptr<sh::filesystem::FilesystemNode> **node** ()

Returns the file node which is later to be opened.

QList<std::shared\_ptr<tools::filetypes::OpenMethod>> **openMethods** ()

Returns the open-methods (programs for opening the file) to present for choice.

qint64 **dialogId** ()

Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

bool **isInited** ()

Returns if this dialog is initialized.

Must be called in main thread.

bool **wasAccepted** ()

Returns if this dialog was 'accepted' by the user (typically by clicking some 'OK' button).

Must be called in main thread.

void **waitClosed** ()

Wait until the user closed the dialog in some way.

May be called in any thread.

void **close** ()

Closes the dialog.

Must be called in main thread.

bool **wasClosed** ()

Returns if this dialog was closed.

Must be called in main thread.

DialogManager \***manager** ()

Returns the *DialogManager* which hosts this dialog.

### Private Slots

```
void cmd__check_another_ancestor__M(WebServerEngineRequest *request)
void cmd__check_custom_opener__M(WebServerEngineRequest *request)
void cmd__open_method_icon__M(WebServerEngineRequest *request)
void cmd__open_methods__M(WebServerEngineRequest *request)
```

## 10.1.460 Class sh::ui::web::WebServerEngine

**class** *sh::ui::web::WebServerEngine*

Abstract base class for a web server engine.

Such an engine implements an http server which a browser can connect to.

### Public Functions

**WebServerEngine** (bool *withMainModule*, int *minport*, int *maxport*, QString *externalUrl*, QString *dispatcherUrl*, QByteArray *wtoken*)

**~WebServerEngine** ()

QUrl **url** () = 0

Returns the home url of this engine (where the end user can point the browser to). .

QUrl **externalUrl** ()

Returns the external root url for creating full externally valid urls (even behind reverse proxies).

Returns *url()* if no external root url is specified.

Change this by the cmdline parameter '*externalUrl*'.

QUrl **dispatcherUrl** ()

Returns the internal root url of the dispatcher (workers only).

QByteArray **dispatcherWtoken** ()

Returns the wtoken got from the dispatcher (workers only).

QUrl **getFullCommandUrl** (QString *command*, *WebCommandData* *data*, bool *external*)

Generates a url path for a command.

Typically only needed for full url generation on server side.

void **triggerEvent** (QString *name*, QJsonObject *data*)

Triggers an event (at the clients).

void **triggerEvent** (QString *name*, QJsonValue *data* = QJsonValue::Undefined)

QString **getConfigValue** (QString *key*)

Returns the value for a runtime configuration key.

void **setConfigValue** (QString *key*, QString *value*)

Sets the runtime configuration key to a value (triggering a event updating the clients).

void **getStaticFile** (*WebServerEngineRequest* \**request*, QString *path*)

Returns a static file content (for the app itself) on a request.

QUrl **rebaseUrl** (QUrl *rootUrl*, QUrl *url*)

Returns a url rebased to another engine's root url.

Used for dispatching to workers.



void **addAndPreserveModule** (std::shared\_ptr<sh::ui::web::WebModule> module)  
 Plugs a *WebModule* into the engine and holds a pointer to it. Unplug it by calling *removeModule()*.

void **addModule** (std::shared\_ptr<sh::ui::web::WebModule> module)  
 Plugs a *WebModule* into the engine. Unplug it by either delete module (drop all pointers to it) or by calling *removeModule()*.

void **removeModule** (sh::ui::web::WebModule \*module)  
 Unplugs a *WebModule* from the engine.

qint64 **currentTimestamp** ()  
 Returns the current engine timestamp.  
 It's not related to real time, but just a value which is higher each time you query it.  
 This is used for coordination of some time-order sensitive operations on browser side.

## Public Static Functions

*WebServerEngine* \***createDispatcherEngine** ()  
 Creates and returns a web engine for a dispatcher.

*WebServerEngine* \***createWorkerEngine** (QString dispatcherUrl, QByteArray wtoken)  
 Creates and returns a web engine for a worker.

### Parameters

- dispatcherUrl: The root url of the dispatcher.

## Private Members

const QJsonObject **\_jok**  
 QMutex **\_configValuesMutex**  
 QMutex **\_modulesMutex**  
 int **\_minport**  
 int **\_maxport**  
 QUrl **\_externalurl**  
 QUrl **\_dispatcherurl**  
 QMap<QString, QString> **\_configValues**  
 QList<std::weak\_ptr<sh::ui::web::WebModule>> **\_modules**  
 QList<std::shared\_ptr<sh::ui::web::WebModule>> **\_smodules**  
 std::shared\_ptr<sh::ui::web::WebServerEngineMainModule> **\_mainmodule**  
 QString **\_webstaticpath**  
 qint64 **\_currentTimestamp** = 1  
 QByteArray **\_wtoken**

## Private Static Functions

*WebServerEngine* \*\_createEngine (bool withMainModule, QString dispatcherUrl, QByteArray wtoken)

## Friends

**friend class** WebServerEngineMainModule

## 10.1.461 Class sh::ui::web::WebServerEngineDispatcher

**class** *sh::ui::web::WebServerEngineDispatcher* : public QObject, public *sh::ui::web::WebModule*

The web serve engine dispatcher module, used in dispatcher mode for providing a portal url which starts Shallot instances on request and internally redirects to them.

## Public Functions

**WebServerEngineDispatcher** ()

**~WebServerEngineDispatcher** ()

void **stop** ()

Stops the dispatcher.

void **setEngine** (*WebServerEngine* \*webserver)

Assigns this web module to a *sh::ui::web::WebServerEngine*. .

*WebServerEngine* \***webserver** ()

Returns the *sh::ui::web::WebServerEngine* this web module is assigned to (may be nullptr).

## Public Static Functions

*WebMainWindow* \***handleDispatching** (std::function<*WebMainWindow*\*) QString dispatcherUrl, QByteArray wtoken  
> *createWndFct* Handles dispatching and creates a new main window (in child processes) on demand via a given factory function.

void **doInitialize** ()

void **doShutdown** ()

bool **executeCommand\_byQObjectReflection** (QObject \*o, *WebServerEngineRequest* \*request, QString command, QString justLeftside = QString())

Convenience function often used by *executeCommand()*.

For a QObject, it searches there for a slot with the name cmd\_{command}, with `/'s replaced by '\_'s (so, for the command "foo/getBars", it searches for slotcmd\_foo\_getBars). If it finds one, it gets executed (in a worker thread). The request is considered as handled, if so.

If there is such a slot, but with appended \_\_M in name, it is called in main thread!

**Return** If it handled (answered) the command.

## Private Functions

bool **compareHashRedirectorWebworkerUrl** (QByteArray *hash*, QString *url*, QString *wtoken*)  
Compares a hash for a redirector webworker url to other data.

*WorkerProcess* \***spawnNewWorker** (*WebServerEngineRequest* \**request*)  
Spawns a new worker.

Does not execute any requests on it.

*WorkerProcess* \***findWorkerForWtoken** (QString *wtoken*)  
Returns the worker for a wtoken.

QString **wtokenFromRequestCookie** (*WebServerEngineRequest* \**request*)  
Returns the wtoken from a request (typically from cookie).

bool **executeCommand** (*WebServerEngineRequest* \**request*, QString *command*) **override**  
Executes *command* as requested by the browser, if responsible.

Read parameters, check if this module is responsible for answering, and then write an answer with *request*.

Implementations often use *executeCommand\_byQObjectReflection()* for convenience.

**Return** If it handled (answered) the command.

bool **rootCommand** (*WebServerEngineRequest* \**request*, QString *webadapter*) **override**  
Returns the root ('index.html') content. .

## Private Members

std::shared\_ptr<*WebServerEngine*> **\_webserver**

QMutex **\_workersmutex**

QHash<QString, *WorkerProcess*\*> **\_workers**

*JanitorThread* **\_janitor**

## Private Slots

void **cmd\_\_drop\_worker\_\_M** (*WebServerEngineRequest* \**request*)

void **cmd\_\_set\_last\_user\_activity** (*WebServerEngineRequest* \**request*)

void **cmd\_\_set\_prevent\_close\_message** (*WebServerEngineRequest* \**request*)

void **cmd\_\_working\_for** (*WebServerEngineRequest* \**request*)

## Private Static Functions

void **start** (QUrl &*url*)  
Starts the dispatcher engine, accepting http connections and internally handling child processes.

QByteArray **hashRedirectorWebworkerUrl** (QString *url*, QString *wtoken*)  
Compute a hash for a redirector webworker url.

### Private Static Attributes

```
std::shared_ptr<WebServerEngineDispatcher> _dispatcher = nullptr
QString _dispatcherId = QString::fromLatin1(sh::tools::Misc::generateUniqueHash())
QString _dispatcherCookieName = QUrl::toPercentEncoding(QString("shallot_%1").arg(_dispatcherId).toLocal8Bit())
class JanitorThread : public QThread
```

### Public Functions

```
JanitorThread (WebServerEngineDispatcher *dispatcher)
void run () override
```

### Private Functions

```
qint64 machineMemory ()
    Returns the amount of memory this machine provides (in bytes).
    Returns -1 if this is not supported on the target platform or it failed.
void stopWorker (WorkerProcess *worker, QString reason)
int detectGoodMaxNumberWorkersByMachineCapabilities ()
QList<WorkerProcess*> stopWorkers (QList<WorkerProcess*> workers, int stopcount, QString
    reason)
```

### Private Members

```
WebServerEngineDispatcher *_dispatcher
double _allMemory
QHash<WorkerProcess*, double> _workerMemoryUsagesLastSeconds
class WorkerProcess : public QProcess
    A internal Shallot worker process driving one Shallot session.
```

### Public Functions

```
QByteArray wtoken ()
    The wtoken (used for association and security).
QUrl rootUrl ()
    The worker root url.
void request (WebServerEngineRequest *request)
    Make a request to the worker.
qint64 workerPid ()
    Returns the process id of the worker.
qint64 memoryUsage ()
    Returns the current memory usage of this worker (in bytes).
    Returns -1 if this is not supported on the target platform or it failed.
```

QDateTime **lastBrowserHeartbeat** ()

Returns when a browser was seen connected to this worker last time.

QDateTime **lastUserInteraction** ()

Returns when a user has interacted with this worker last time.

QString **clientHost** ()

Returns the client host this worker is serving.

It's not guaranteed to have some specific content (maybe an ip address), but is equal for the same host, some one can count how many hosts work for one particular client host.

QString **preventCloseMessage** ()

Returns a reason message why this worker currently should not be closed (or "" if closing is fine).

See also `isClosingInappropriate()`.

Note: This is reported asynchronously. You must not rely on its precise information, but solely use it for monitoring.

bool **isClosingFine** ()

Returns if it is fine to stop this worker (or if there is a good reason to not do).

See also `preventCloseMessage()`.

Note: This is reported asynchronously. You must not rely on its precise information, but solely use it for monitoring.

## Private Functions

**WorkerProcess** (*WebServerEngineDispatcher* \*dispatcher, QByteArray wtoken, QString clientHost, QStringList acceptedLanguages)

bool **waitOnline** ()

Waits until the worker is online and ready for requests (or dead).

void **initialize** (QString rooturl, quint64 pid)

Initializes the worker instance (which exists on the dispatcher side!) by setting some final data.

void **gotBrowserHeartbeat** ()

Refreshes the browser heartbeat timestamp.

See `lastBrowserHeartbeat()`.

void **gotUserInteraction** (int value)

Refreshes the user interaction timestamp.

See `lastUserInteraction()`.

void **setPreventCloseMessage** (QString message)

Sets the preventCloseMessage (empty: closeable).

### Private Members

```
WebServerEngineDispatcher *_dispatcher
QByteArray _wtoken
QUrl _rooturl
QMutex _mutex
QWaitCondition _cnd_initied
qint64 _pid = -1
QDateTime _lastBrowserHeartbeat
QDateTime _lastUserInteraction
QString _clientHost
QString _preventCloseMessage
QRegularExpression _reProcStatus
```

### Friends

```
friend class WebServerEngineDispatcher
```

## 10.1.462 Class `sh::ui::web::WebServerEngineDispatcher::JanitorThread`

```
class sh::ui::web::WebServerEngineDispatcher::JanitorThread: public QThread
```

### Public Functions

```
JanitorThread(WebServerEngineDispatcher *dispatcher)
void run() override
```

### Private Functions

```
qint64 machineMemory()
    Returns the amount of memory this machine provides (in bytes).
    Returns -1 if this is not supported on the target platform or it failed.
void stopWorker(WorkerProcess *worker, QString reason)
int detectGoodMaxNumberWorkersByMachineCapabilities()
QList<WorkerProcess*> stopWorkers(QList<WorkerProcess*> workers, int stopcount, QString rea-
    son)
```

## Private Members

*WebServerEngineDispatcher* \*\_dispatcher

double \_allMemory

QHash<*WorkerProcess*\*, double> \_workerMemoryUsagesLastSeconds

### 10.1.463 Class sh::ui::web::WebServerEngineDispatcher::WorkerProcess

**class** *sh::ui::web::WebServerEngineDispatcher::WorkerProcess* : **public** QProcess

A internal Shallot worker process driving one Shallot session.

## Public Functions

QByteArray **wtoken** ()

The wtoken (used for association and security).

QUrl **rootUrl** ()

The worker root url.

void **request** (*WebServerEngineRequest* \*request)

Make a request to the worker.

qint64 **workerPid** ()

Returns the process id of the worker.

qint64 **memoryUsage** ()

Returns the current memory usage of this worker (in bytes).

Returns -1 if this is not supported on the target platform or it failed.

QDateTime **lastBrowserHeartbeat** ()

Returns when a browser was seen connected to this worker last time.

QDateTime **lastUserInteraction** ()

Returns when a user has interacted with this worker last time.

QString **clientHost** ()

Returns the client host this worker is serving.

It's not guaranteed to have some specific content (maybe an ip address), but is equal for the same host, some one can count how many hosts work for one particular client host.

QString **preventCloseMessage** ()

Returns a reason message why this worker currently should not be closed (or "" if closing is fine).

See also *isClosingInappropriate()*.

Note: This is reported asynchronously. You must not rely on its precise information, but solely use it for monitoring.

bool **isClosingFine** ()

Returns if it is fine to stop this worker (or if there is a good reason to not do).

See also *preventCloseMessage()*.

Note: This is reported asynchronously. You must not rely on its precise information, but solely use it for monitoring.

## Private Functions

**WorkerProcess** (*WebServerEngineDispatcher* \*dispatcher, QByteArray wtoken, QString clientHost, QStringList acceptedLanguages)

bool **waitOnline** ()

Waits until the worker is online and ready for requests (or dead).

void **initialize** (QString rooturl, qint64 pid)

Initializes the worker instance (which exists on the dispatcher side!) by setting some final data.

void **gotBrowserHeartbeat** ()

Refreshes the browser heartbeat timestamp.

See *lastBrowserHeartbeat()*.

void **gotUserInteraction** (int value)

Refreshes the user interaction timestamp.

See *lastUserInteraction()*.

void **setPreventCloseMessage** (QString message)

Sets the preventCloseMessage (empty: closeable).

## Private Members

*WebServerEngineDispatcher* \*\_dispatcher

QByteArray \_wtoken

QUrl \_rooturl

QMutex \_mutex

QWaitCondition \_cnd\_initied

qint64 \_pid = -1

QDateTime \_lastBrowserHeartbeat

QDateTime \_lastUserInteraction

QString \_clientHost

QString \_preventCloseMessage

QRegularExpression \_reProcStatus

## Friends

**friend class** WebServerEngineDispatcher



### 10.1.464 Class `sh::ui::web::WebServerEngineMainModule`

**class** `sh::ui::web::WebServerEngineMainModule` : **public** `QObject`, **public** `sh::ui::web::WebModule`  
 The web serve engine main module, providing some base services like events.

#### Public Functions

**WebServerEngineMainModule** ()

**~WebServerEngineMainModule** ()

**bool executeCommand** (*WebServerEngineRequest* \*request, *QString* command) **override**  
 Executes command as requested by the browser, if responsible.

Read parameters, check if this module is responsible for answering, and then write an answer with request.

Implementations often use *executeCommand\_byQObjectReflection()* for convenience.

**Return** If it handled (answered) the command.

**bool rootCommand** (*WebServerEngineRequest* \*request, *QString* webadapter) **override**  
 Returns the root ('index.html') content. .

**void triggerEvent** (*QString* name, *QString* data)  
 Triggers an event on browser side.

**void setEngine** (*WebServerEngine* \*webserver)  
 Assigns this web module to a *sh::ui::web::WebServerEngine*. .

*WebServerEngine* \*webserver ()  
 Returns the *sh::ui::web::WebServerEngine* this web module is assigned to (may be nullptr).

#### Public Static Functions

**bool executeCommand\_byQObjectReflection** (*QObject* \*o, *WebServerEngineRequest* \*request, *QString* command, *QString* justLeftside = *QString*())

Convenience function often used by *executeCommand()*.

For a *QObject*, it searches there for a slot with the name `cmd_{command}`, with ``'s replaced by '\_'s (so, for the command "foo/get\_bars", it searches for `slotcmd_foo_get_bars``). If it finds one, it gets executed (in a worker thread). The request is considered as handled, if so.

If there is such a slot, but with appended `__M` in name, it is called in main thread!

**Return** If it handled (answered) the command.

### Private Functions

qint64 **lastRequestTime** ()

### Private Members

qint64 **\_lastRequestTime**

QMutex **\_mutex\_lastRequestTime**

QList<*EventEntry*\*> **\_eventEntries**

qint64 **\_eventEntriesLastId** = 0

QMutex **\_mutex\_eventEntries**

QWaitCondition **\_eventEntriesCondition**

*EventEntryCleanupThread* **\_eventEntryCleanupThread**

*StopWhenIdleThread* **\_stopWhenIdleThread**

### Private Slots

void **cmd\_dispatcher\_toclient\_heartbeat** (*WebServerEngineRequest* \*request)

void **cmd\_events\_get\_next** (*WebServerEngineRequest* \*request)

### Private Static Attributes

QRegularExpression **\_restatic**

**class EventEntry**

Data for one occurred event.

### Public Functions

**EventEntry** (qint64 *id*, qint64 *time*, QString *name*, QString *data*)

### Public Members

**const** qint64 **id**

**const** qint64 **time**

**const** QString **name**

**const** QString **data**

### Friends

```
friend class EventEntryCleanupThread

class EventEntryCleanupThread: public QThread
    Thread for cleaning up old EventEntry instances.
```

### Public Functions

```
EventEntryCleanupThread(WebServerEngineMainModule *module)
```

### Private Members

```
WebServerEngineMainModule *module

class StopWhenIdleThread: public QThread
    Thread for stopping the application if the remote side is gone.
```

### Public Functions

```
StopWhenIdleThread(WebServerEngineMainModule *module)
```

### Private Members

```
WebServerEngineMainModule *module
```

## 10.1.465 Class sh::ui::web::WebServerEngineMainModule::EventEntry

```
class sh::ui::web::WebServerEngineMainModule::EventEntry
    Data for one occurred event.
```

### Public Functions

```
EventEntry(qint64 id, qint64 time, QString name, QString data)
```

### Public Members

```
const qint64 id
const qint64 time
const QString name
const QString data
```

## Friends

**friend class** EventEntryCleanupThread

### 10.1.466 Class sh::ui::web::WebServerEngineMainModule::EventEntryCleanupThread

**class** *sh::ui::web::WebServerEngineMainModule::EventEntryCleanupThread* : **public** QThread  
Thread for cleaning up old *EventEntry* instances.

## Public Functions

**EventEntryCleanupThread** (*WebServerEngineMainModule* \*module)

## Private Members

*WebServerEngineMainModule* \*module

### 10.1.467 Class sh::ui::web::WebServerEngineMainModule::StopWhenIdleThread

**class** *sh::ui::web::WebServerEngineMainModule::StopWhenIdleThread* : **public** QThread  
Thread for stopping the application if the remote side is gone.

## Public Functions

**StopWhenIdleThread** (*WebServerEngineMainModule* \*module)

## Private Members

*WebServerEngineMainModule* \*module

### 10.1.468 Class sh::ui::web::WebServerEngineRequest

**class** *sh::ui::web::WebServerEngineRequest*

A web request in a *WebServerEngine*.

Override this together with *WebServerEngine*.

## Public Functions

**WebServerEngineRequest** (QString *url*, QString *clientHost*)

QUrl **url** ()

Returns the requested url.

Note: The url should be considered as opaque, since the exact structure depends on the engine implementation.

QString **clientHost** ()

Returns the client host address.

QString **path** ()

Returns the path part of the requested url.

Note: The url should be considered as opaque, since the exact structure depends on the engine implementation.

QString **data** (QString *key*)

Returns data passed as parameter along with the request by *key*.

int **responseCode** ()

Returns the answered (http) response code.

QString **responseMimeType** ()

Returns the mimetype of the answer content.

QByteArray **response** ()

Returns the answer content.

bool **responseSet** ()

Returns if a response was set.

void **setResponse** (QByteArray *response*, QString *mimetype*, int *responseCode* = *HTTP\_OK*)

Sets the response.

void **setResponse** (QJsonArray *response*, int *responseCode* = *HTTP\_OK*)

void **setResponse** (QJsonObject *response*, int *responseCode* = *HTTP\_OK*)

QString **headerData** (QString *key*) = 0

Returns an http header value by key. .

QString **getCookie** (QString *key*) = 0

Returns an http cookie stored on browser side.

Note: This only works on dispatcher level and not in usual *WebModule* commands.

void **setCookie** (QString *key*, QString *value*, int *maxAgeSecs* = -1) = 0

Sets an http cookie to be returned to the browser side.

Note: This only works on dispatcher level and not in usual *WebModule* commands.

## Public Static Attributes

const int **HTTP\_OK** = 200

const int **HTTP\_SEE\_OTHER** = 303

const int **HTTP\_NOT\_FOUND** = 404

const int **HTTP\_INTERNAL\_SERVER\_ERROR** = 500

const int **HTTP\_BAD\_GATEWAY** = 502

### Private Members

```
QUrl _url
QString _clientHost
QString _path
QMap<QString, QString> _getdata
int _responseCode = HTTP_NOT_FOUND
QString _responseMimeType = "application/octet-stream"
QByteArray _response
bool _responseSet = false
```

## 10.1.469 Class sh::ui::web::WebStoreProfileDialog

```
class sh::ui::web::WebStoreProfileDialog : public QObject, public sh::ui::web::WebDialog, public sh::ui::StoreProfileDialog {
    Web based save settings dialog.
```

### Public Functions

```
WebStoreProfileDialog (std::shared_ptr<const sh::filesystem::Eurl> eurl)
```

```
QList<SettingEntry> checkedSettings () override
    Returns the list of settings the user has checked to store.
```

```
QString profileName () override
    Returns the profile the user has chosen to store settings for.
```

```
bool withSubDirectories () override
    Returns if the user has chosen to apply the checked settings also for subdirectories.
```

```
QStringList inheritsFrom () override
    Returns the list of profiles the user has chosen to inherit from.
```

```
bool hasGlobal () override
    Returns if the user has chosen to apply the checked settings for each directory (instead of the current directory).
```

```
QVariant dialogProperty (QString dlgPropertyKey, QVariant deflt = QVariant())
    Returns a dialog property value by key.
```

```
void setDialogProperty (QString dlgPropertyKey, QVariant value)
    Sets a dialog property value for a key.
```

```
QVariantHash dialogResult ()
    Returns the dialog result as hash.
```

The browser side of a web dialog can ‘accept’ a dialog while closing by setting a non-empty dialog result. This mechanism allows browser side dialogs to return data to the backend. It’s up to the backend to interpret the content of this hash.

If the dialog was cancelled, or is still open, the returned hash is empty. See also wasClosed().

```
void triggerDialogEvent (QString name, QJsonObject data = QJsonObject())
    Triggers a dialog event (at the browsers).
```

They in turn typically fetch some data again in order to refresh the ui.

void **setCloseableByUser** (bool v = true)  
 Sets if this dialog shall be directly closeable by the user by the window decoration. Set to `false` if the user has to react on this dialog by clicking on some controls in the dialog body.

This method must be called during construction.

bool **isCloseableByUser** ()  
 Returns if this dialog shall be directly closeable by the user by the window decoration.

QJsonObject **toJson** () **override**  
 Returns the json representation as QJsonObject.

std::shared\_ptr<const *sh::filesystem::Eurl*> **eurl** ()  
 Returns the current directory this dialog shall work on.

qint64 **dialogId** ()  
 Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

bool **isInited** ()  
 Returns if this dialog is initialized.

Must be called in main thread.

bool **wasAccepted** ()  
 Returns if this dialog was ‘accepted’ by the user (typically by clicking some ‘OK’ button).

Must be called in main thread.

void **waitClosed** ()  
 Wait until the user closed the dialog in some way.

May be called in any thread.

void **close** ()  
 Closes the dialog.

Must be called in main thread.

bool **wasClosed** ()  
 Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \***manager** ()  
 Returns the *DialogManager* which hosts this dialog.

### Private Slots

void **cmd\_\_get\_\_M** (*WebServerEngineRequest* \*request)  
 void **cmd\_\_store\_mode\_\_M** (*WebServerEngineRequest* \*request)

## 10.1.470 Class `sh::ui::web::WebTuningDialog`

**class** `sh::ui::web::WebTuningDialog` : **public** `QObject`, **public** `sh::ui::web::WebDialog`, **public** `sh::ui::TuningDialog`  
Web based tuning dialog.

### Public Functions

**WebTuningDialog** ()

`QVariant dialogProperty` (`QString dlgPropertyKey`, `QVariant deflt` = `QVariant()`)  
Returns a dialog property value by key.

**void setDialogProperty** (`QString dlgPropertyKey`, `QVariant value`)  
Sets a dialog property value for a key.

`QVariantHash dialogResult` ()  
Returns the dialog result as hash.

The browser side of a web dialog can ‘accept’ a dialog while closing by setting a non-empty dialog result. This mechanism allows browser side dialogs to return data to the backend. It’s up to the backend to interpret the content of this hash.

If the dialog was cancelled, or is still open, the returned hash is empty. See also `wasClosed()`.

**void triggerDialogEvent** (`QString name`, `QJsonObject data` = `QJsonObject()`)  
Triggers a dialog event (at the browsers).

They in turn typically fetch some data again in order to refresh the ui.

**void setCloseableByUser** (`bool v` = `true`)  
Sets if this dialog shall be directly closeable by the user by the window decoration. Set to `false` if the user has to react on this dialog by clicking on some controls in the dialog body.

This method must be called during construction.

**bool isCloseableByUser** ()  
Returns if this dialog shall be directly closeable by the user by the window decoration.

`QJsonObject toJson` () **override**  
Returns the json representation as `QJsonObject`.

**qint64 dialogId** ()  
Returns the dialog id.  
  
Each instance has an id unique in the complete Shallot process lifetime.  
  
Must be called in main thread.

**bool isInited** ()  
Returns if this dialog is initialized.  
  
Must be called in main thread.

**bool wasAccepted** ()  
Returns if this dialog was ‘accepted’ by the user (typically by clicking some ‘OK’ button).  
  
Must be called in main thread.

**void waitClosed** ()  
Wait until the user closed the dialog in some way.  
  
May be called in any thread.



void **close** ()  
 Closes the dialog.  
 Must be called in main thread.

bool **wasClosed** ()  
 Returns if this dialog was closed.  
 Must be called in main thread.

*DialogManager* \***manager** ()  
 Returns the *DialogManager* which hosts this dialog.

### Private Members

QMutex **\_cfgvaluesmutex**  
 QWaitCondition **\_cfgvaluescondition**  
 bool **\_cfgvaluesinitd** = false  
 QList<std::shared\_ptr<*sh::configuration::ConfigurationValue*>> **\_cfgvalues**

### Private Slots

void **cmd\_\_change\_configurationvalue** (*WebServerEngineRequest* \*request)  
 void **cmd\_\_get\_configurationvalues** (*WebServerEngineRequest* \*request)

### Private Static Functions

QString **categoryToName** (*sh::configuration::ConfigurationCategory* category)  
 QString **valueTypeToName** (*sh::configuration::ConfigurationValueType* \*valueType)  
 QJsonObject **configurationValueToJsonObject** (std::shared\_ptr<*sh::configuration::ConfigurationValue*> cfgval)

## 10.1.471 Class **\_ProjectInfo**

**class \_ProjectInfo**  
 Some internal program information.  
 This is auto-generated content only used by the core.

### Public Members

**const** char \***version** = "1.2.4767"  
**const** char \***builddtimeutc** = "2020-07-26 16:33:11"  
**const** char \***homepage** = "https://pseudopolis.eu/wiki/pino/projs/shallot"

## 10.2 Namespace list

### 10.2.1 Namespace Ui

**namespace Ui**  
Qt-internal for ui.

### 10.2.2 Namespace sh

**namespace sh**  
Shallot root.

**class MainInit**  
(Un)Initialization for main app.

#### Public Static Functions

void **doInitialize** ()  
void **doShutdown** ()

#### Public Static Attributes

*sh::base::SingletonInitializer* \***singletonInitializer** = nullptr  
sig\_atomic\_t **caughtSigterm** = 0

#### namespace actions

Infrastructure for actions (submenus, executable ones, and more), as shown in the toolbar and menus and used at some other places.

See the abstract base class *sh::actions::AbstractActionItem* for more.

#### class AbstractActionFactory

*#include <actionfactory.h>* Abstract base class for an action factory.

It creates an instance of a certain subclass of *AbstractActionItem*. It can also check if it is defined to be created in a given situation (e.g. not all actions are visible in toolbar).

Subclassed by *sh::actions::ActionFactory< T >*, *sh::scripting::api::ApiActionFactory*

#### Public Functions

**AbstractActionFactory** (std::shared\_ptr<*sh::actions::ActionCategory*> *category*,  
QList<std::shared\_ptr<*Predicate*>> *predicates*)  
Is (for subclasses) intended to be directly constructed and registered once.

**~AbstractActionFactory** ()

std::shared\_ptr<*ActionInstantiation*> **actionAvailable** (*ActionInstantiation* \**instantia-*  
*tion*)

std::shared\_ptr<*sh::actions::AbstractActionItem*> **construct** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>  
*nodes*) = 0

## Private Members

`std::shared_ptr<sh::actions::ActionCategory> category`

`QList<std::shared_ptr<Predicate>> predicates`

**class AbstractActionItem**: public QObject, public std::enable\_shared\_from\_this<AbstractActionItem>  
*#include <abstractactionitem.h>* Abstract base class for executable actions, submenus of them, and more.

See the subclasses of this class.

They are used at various places. The toolbar and context menus provide them to the user.

They can be registered e.g. in a *sh::actions::ActionFactory* or returned from *sh::filesystem::FilesystemHandler::getActions* in order to make them available.

Subclassed by *sh::actions::ActionActionItem*, *sh::actions::HeaderActionItem*, *sh::actions::SeparatorActionItem*, *sh::actions::SubmenuActionItem*

## Public Functions

QString **text** ()

Returns the displayed text for this action.

QString **icon** ()

Returns the icon for this action.

bool **enabled** ()

Checks if this action is enabled.

bool **isChecked** ()

Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () const

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See *ActionDefaultPrecedenceValues* for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

`std::weak_ptr<AbstractActionItem> parentAction ()`  
Returns the parent action, if it is added to a container.

`void _setParentAction (std::shared_ptr<AbstractActionItem> parent)`  
Sets the parent action. .

`void initializeAsync (std::function<void>  
> oninitialized = 0)` Asynchronously initializes the action.

`void initializeSync ()`  
Synchronously initializes the action.

`bool isInitialized ()`  
Checks if this action is initialized.

`bool isInitializing ()`  
Checks if this action is initializing.

`AbstractActionItem ()`

### Signals

`void changed ()`  
Emits when some data changed.

### Private Members

`QString _text`  
`QString _icon`  
`bool _enabled`  
`bool _visible`  
`bool _ischeckable`  
`bool _ischecked`  
`bool _initialized`  
`bool _initializing`  
`bool _reinitialize`  
`std::weak_ptr<AbstractActionItem> _parentAction = NO_PARENT`  
`int _defaultActionPrecedence`

### Private Static Attributes

`QWaitCondition initcondition`

**class ActionActionItem:** public *sh::actions::AbstractActionItem*  
*#include <actionactionitem.h>* Abstract base class for an executable action.

It can be made visible in menus or the toolbar or executed directly.

Subclasses provide the actual implementation by implementing `action`. Execution can be triggered from outside by calling `execute`.

Subclassed by `sh::actions::common::ActionAbstractTransferTree`,  
`sh::actions::common::ActionAddToBookmarks`, `sh::actions::common::ActionBookmark`,  
`sh::actions::common::ActionClipboardCopy`, `sh::actions::common::ActionClipboardCut`,  
`sh::actions::common::ActionClipboardPaste`, `sh::actions::common::ActionClipboardPasteAsLink`,  
`sh::actions::common::ActionCreateFile`, `sh::actions::common::ActionCreateFolder`,  
`sh::actions::common::ActionDeleteItems`, `sh::actions::common::ActionExecute`,  
`sh::actions::common::ActionManageBookmarks`, `sh::actions::common::ActionNavigateInHistory`,  
`sh::actions::common::ActionOpenArchive`, `sh::actions::common::ActionOpenDirectoryInNewWindow`,  
`sh::actions::common::ActionOpenFile`, `sh::actions::common::ActionOpenFileWith::_ActionOpenFileWithEntry`,  
`sh::actions::common::ActionOpenFileWith::_ActionOpenFileWithUI`,  
`sh::actions::common::ActionOpenSharcArchive`, `sh::actions::common::ActionOpenTerminalAsUser`,  
`sh::actions::common::ActionRenameItem`, `sh::actions::common::ActionShowProperties`,  
`sh::actions::common::ActionTransferToHelper`, `sh::actions::mainmenu::ActionAbout`,  
`sh::actions::mainmenu::ActionAbstractSelectNodes`, `sh::actions::mainmenu::ActionApplyProfile::ActionApplyProfileX`,  
`sh::actions::mainmenu::ActionFileViewChangeIconDimension`, `sh::actions::mainmenu::ActionFileviewFilesizeFormat`,  
`sh::actions::mainmenu::ActionFileviewShowHiddenFiles`, `sh::actions::mainmenu::ActionFileviewView`,  
`sh::actions::mainmenu::ActionGotoDirectory`, `sh::actions::mainmenu::ActionHelp`,  
`sh::actions::mainmenu::ActionManageSavedSettings`, `sh::actions::mainmenu::ActionNumberFileviews_Add`,  
`sh::actions::mainmenu::ActionNumberFileviews_Remove`, `sh::actions::mainmenu::ActionNumberFileviews_Remove_C`,  
`sh::actions::mainmenu::ActionQuit`, `sh::actions::mainmenu::ActionRefresh`,  
`sh::actions::mainmenu::ActionSaveSettings`, `sh::actions::mainmenu::ActionSearch`,  
`sh::actions::mainmenu::ActionSetWindowTitlePattern`, `sh::actions::mainmenu::ActionShowDetailPanel`,  
`sh::actions::mainmenu::ActionShowFolderTree`, `sh::actions::mainmenu::ActionShowLog`,  
`sh::actions::mainmenu::ActionTreeStickyness`, `sh::actions::mainmenu::ActionTuning`,  
`sh::filepropertydialogtabs::FilePropertyDialogTabExtendedAttributes::ActionAddAttribute`,  
`sh::filepropertydialogtabs::FilePropertyDialogTabExtendedAttributes::ActionChangeAttribute`,  
`sh::filepropertydialogtabs::FilePropertyDialogTabExtendedAttributes::ActionRemoveAttribute`,  
`sh::filepropertydialogtabs::FilePropertyDialogTabUnixPermissions::ActionChange`,  
`sh::filesystemhandlers::ActionMountGnomeIOLocation`, `sh::filesystemhandlers::ActionMountNetworkGnomeIOLocation`,  
`sh::filesystemhandlers::ActionUnmountGnomeIOLocation`, `sh::scripting::api::ApiActionActionItem`,  
`sh::scripting::ScriptingEngine::ActionPluginLoadCrashedAgain`, `sh::search::criteria::ActionAbstractActionDrivenSea`

## Public Functions

**ActionActionItem** (QString *text*, bool *enabled* = true, QString *icon* = QString(), int *default-ActionPrecedence* = 0, bool *checkable* = false, bool *ischecked* = false)  
 Is (for subclasses) intended to be directly constructed from everywhere or by registering a factory somewhere.

void **execute** ()  
 Executes this action.

void **execute** (sh::actions::ActionExecutionInfo \**info*)  
 Executes this action.

QKeySequence **shortcuthint** ()  
 Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
 Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
 Sets the keyboard shortcut for triggering this action.

**QString text ()**  
Returns the displayed text for this action.

**QString icon ()**  
Returns the icon for this action.

**bool enabled ()**  
Checks if this action is enabled.

**bool isChecked ()**  
Checks if this action is checked (has a cross in the ui).

**bool isCheckedable ()**  
Checks if this action is checkable (can have a cross in the ui).

**int defaultActionPrecedence () const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

**void setText (QString text)**  
Sets the displayed text.

**void setIcon (QString icon)**  
Sets the icon.

**void setEnabled (bool enabled)**  
Sets if the item is enabled.

**void setChecked (bool checked)**  
Sets if the item is checked (has a cross in the ui).

**void setVisible (bool visible)**  
Sets the visibility of this item.

**bool visible ()**  
Checks the visibility of this item (non-recursively).

**std::weak\_ptr<AbstractActionItem> parentAction ()**  
Returns the parent action, if it is added to a container.

**void \_setParentAction (std::shared\_ptr<AbstractActionItem> parent)**  
Sets the parent action. .

**void initializeAsync (std::function<void>  
> oninitialized = 0)**Asynchronously initializes the action.

**void initializeSync ()**  
Synchronously initializes the action.

**bool isInitialized ()**  
Checks if this action is initialized.

**bool isInitializing ()**  
Checks if this action is initializing.

## Signals

void **changed** ()  
 Emits when some data changed.

## class **ActionCategory**

*#include <actioncategory.h>* Action categories are used for grouping actions in menus.

They provide the primary grouping in the toolbar an the context menus. Per default, the categories "create", "open" and "manage" exist.

## Public Functions

**ActionCategory** (QString *name*, QString *label*, QString *icon*)  
 Constructed only by the infrastructure and made available otherwise.

QString **name** ()  
 The category name.

QString **label** ()  
 The category label.

QString **icon** ()  
 The category icon.

## Public Static Functions

std::shared\_ptr<*ActionCategory*> **getByName** (QString *name*)  
 Gets a category by name.

std::shared\_ptr<*ActionCategory*> **add** (QString *name*, QString *label*, QString *icon*)  
 Adds a new category.

QList<std::shared\_ptr<*ActionCategory*>> **categories** ()  
 Returns a list of all existing categories.

void **doInitialize** ()

void **doShutdown** ()

## Private Members

QString **\_name**

QString **\_label**

QString **\_icon**

### Private Static Attributes

QList<std::shared\_ptr<ActionCategory>> \_categories

#### class ActionDefaultPrecedenceValues

#include <abstractactionitem.h> Reference values for calculating default precedence values of actions.

### Public Static Attributes

const int PRECEDENCE\_OPENFILE = 300000

Normal file open action.

const int PRECEDENCE\_BUILTIN\_SPECIAL\_OPEN = 600000

Special open action.

#### class ActionExecutionInfo : public QObject

#include <actionexecutioninfo.h> Programming interface for giving status infos and communication with the user in an action execution.

An instance is always available during an execution.

### Public Functions

ActionExecutionInfo (QObject \*parent = 0)

Constructed only by the infrastructure and made available otherwise.

~ActionExecutionInfo ()

void setVisualProcessFeedbackActive (bool active)

Specifies if there must be a visual process feedback.

The visual process feedback is typically realized as a dialog showing progress information.

Note: It can also appear in some situations, even if this property is set to false.

bool isVisualProcessFeedbackActive ()

Determines if there must be a visual process feedback.

void setManualInterventionNeeded (bool isneeded)

Specifies if manual input is currently required.

bool isManualInterventionNeeded ()

Determines if manual input is currently required.

void startExecution (std::shared\_ptr<sh::actions::AbstractActionItem> action)

Triggers the beginning of the actual execution. .

void finishExecution ()

Triggers the finishing of the execution. .

void setDetails (QString fromverb, QString fromobj, QString toverb = QString(), QString toobj = QString())

Sets some detail texts.

QString fromVerb ()

Gets detail text 'from-verb'.

QString fromObjectName ()

Gets detail text 'from-objectname'.



QString **toVerb** ()  
Gets detail text 'to-verb'.

QString **toObjectName** ()  
Gets detail text 'to-objectname'.

void **setHead** (QString *text*)  
Sets the header text.

QString **head** ()  
Gets the header text.

void **setProgressIndeterminate** (QString *text* = QString())  
Sets progress information to 'indeterminate'.

void **setProgress** (quint64 *done*, quint64 *all*, QString *text*)  
Sets progress information.

bool **isProgressDeterminate** ()  
Gets if progress information is 'indeterminate'.

quint64 **progressDone** ()  
Gets count of finished items.

quint64 **progressAll** ()  
Gets count of all items.

QString **progressText** ()  
Gets progress information text.

*sh::actions::ActionExecutionUserFeedback* \***userfeedback** ()  
Program interface for interactive user input.

bool **isCancelled** ()  
Gets if the current action execution was cancelled.

void **cancel** ()  
Cancel the current action execution.  
  
Used from outside, e.g. as handler for a 'Cancel' button.

void **respectCancel** ()  
Called regularly from an action execution code in order to cancel execution at that place, if the user has cancelled it.

void **withExecuteGuards\_infodlg** (std::function<void>  
>int *flags* = 0)Executes code with different exception handlers for presenting exceptions in the action execution inside the action dialog (instead of the default exception dialog).

*sh::filesystem::Operation* \***operation** ()  
Program interface for operations on the filesystem (reading or writing).

void **addChangedEurl** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*)  
Reminds an important change of a certain node in the filesystem (file created, modified, removed?).  
  
The infrastructure will force the ui to refresh those nodes.

void **shutdown** (bool *success*)  
Shuts down this execution info and releases resources.

## Signals

void **detailsChanged** ()  
Signal for changed execution details.

void **headChanged** ()  
Signal for changed head.

void **progressChanged** ()  
Signal for changed progress.

void **visualProcessFeedbackActiveChanged** ()  
Signal for changed visibility-required.

void **wasCancelled** ()  
Signal for cancellation on user behalf.

bool **manualInterventionNeededChanged** ()  
Signal for changed manual-intervention-needed.

## Private Functions

std::shared\_ptr<sh::ui::ActionExecutionInfoDialog> **createDialog** ()

std::shared\_ptr<ui::ActionExecutionInfoPanel> **createPanel** (sh::ui::ActionExecutionInfoDialog  
\*dialog)

## Private Members

QMutex **mutex**

QString **\_head**

QString **\_fromverb**

QString **\_fromobj**

QString **\_toverb**

QString **\_toobj**

bool **\_progressDeterminate**

quint64 **\_progressDone**

quint64 **\_progressAll**

bool **\_visualProcessFeedbackActive** = false

bool **\_wasAlwaysInvisible** = true

bool **\_manualInterventionNeeded** = false

QString **\_progressText**

std::shared\_ptr<sh::ui::ActionExecutionInfoDialog> **infodialog** = 0

std::shared\_ptr<sh::ui::ActionExecutionInfoPanel> **infopanel** = 0

bool **\_iscancelled** = false

sh::filesystem::Operation **\_operation**

QList<std::shared\_ptr<const sh::filesystem::Eurl>> **changes**

QStack<std::shared\_ptr<*sh::actions::AbstractActionItem*>> **stack**

## Private Slots

void **\_checkVisibilityOnce**()

**class ActionExecutionUserFeedback**

*#include <actionexecutionuserfeedback.h>* Methods for user interaction in an action execution.

When an action executes, it may need to ask the user for some input at some time. An action implementation can do so by calling those methods. It is available as a member of the *ActionExecutionInfo* class. An instance of it is available in each action execution.

Subclassed by *sh::ui::ActionExecutionInfoDialog*

## Public Types

**enum MessageBoxButton**

Buttons in a message box from *ActionExecutionUserFeedback*.

*Values:*

**enumerator NONE** = 0

**enumerator OK** = 1 << 0

**enumerator Continue** = 1 << 1

**enumerator Cancel** = 1 << 2

**enumerator Retry** = 1 << 3

**enumerator Yes** = 1 << 4

**enumerator No** = 1 << 5

## Public Functions

int **messageBox**(QString *text*, QList<QString> *answers*, QString *icon* = QString(), int *defaultanswer* = -1, int *cancelanswer* = -1, QList<QString> *answericons* = QList<QString>()) = 0

int **inputBox**(QString *text*, QList<QString> *answers*, QString \**value*, QString *icon* = QString(), int *defaultanswer* = -1, int *cancelanswer* = -1, int *valuePreselectFrom* = -1, int *valuePreselectTo* = -1) = 0

int **multilineInputBox**(QString *text*, QList<QString> *answers*, QString \**value*, QString *icon* = QString(), int *defaultanswer* = -1, int *cancelanswer* = -1) = 0

int **simpleChooserGridform**(QString *text*, *GridformEntries* \**entries*, QStringList *answers*, int *defaultanswer* = -1, int *cancelanswer* = -1) = 0

bool **credentialsDialog**(QString *text*, bool *showDomain*, bool *showUsername*, bool *showPassword*, bool *showAnonymous*, bool *showRemember*, QString \**domain*, QString \**username*, QString \**password*, bool \**isAnonymous*, bool \**isRemember*) = 0

```
bool unixPermissionsDialog (bool *userMayRead, bool *userMayWrite, bool *userMayExecute, bool *groupMayRead, bool *groupMayWrite, bool *groupMayExecute, bool *othersMayRead, bool *othersMayWrite, bool *othersMayExecute, bool *sticky, bool *setuid, bool *setgid, QStringList users, QStringList groups, QString *ownerUser, QString *ownerGroup) = 0
```

```
QString simpleInputDialog (QString text, QString deflt, int valuePreselectFrom = -1, int valuePreselectTo = -1)
```

```
int simpleMessageBox (QString text, int buttons = (int) MessageBoxButton::OK, QString icon = QString(), int defaultbutton = (MessageBoxButton)0, int cancelbutton = (MessageBoxButton)0)
```

```
~ActionExecutionUserFeedback ()
```

```
class Choice
```

*#include <actionexecutionuserfeedback.h>* A data structure for a choice (in an *GridformEntry*).

### Public Functions

```
Choice (QString label, QString text = QString())
```

### Public Members

```
QString label
```

```
QString text
```

```
class Choices
```

*#include <actionexecutionuserfeedback.h>* A data structure for a list of *Choice* instances.

### Public Members

```
QList<Choice*> list
```

```
int selected = -1
```

```
class GridformEntries
```

*#include <actionexecutionuserfeedback.h>* A data structure for a list of entries in a simple chooser grid form.

### Public Functions

```
GridformEntries ()
```

Is intended to be directly constructed from everywhere.

```
~GridformEntries ()
```

```
GridformEntry *newEntry (QString label)
```

## Public Members

QList<*GridformEntry*\*> **list**

**class GridformEntry**

*#include <actionexecutionuserfeedback.h>* A data structure for an entry in a simple chooser grid form.

## Public Functions

**GridformEntry** (QString *label*, *Choices choices*)

Constructed only indirectly.

**~GridformEntry** ()

int **addChoice** (QString *label*, QString *text* = QString())

## Public Members

QString **label**

*Choices choices*

template<class T>

**class ActionFactory** : public *sh::actions::AbstractActionFactory*

*#include <actionfactory.h>* A typical implementation for an action factory.

See base class for more.

## Public Functions

**ActionFactory** (std::shared\_ptr<*sh::actions::ActionCategory*> *category*,  
QList<std::shared\_ptr<*Predicate*>> *predicates*)

std::shared\_ptr<*sh::actions::AbstractActionItem*> **construct** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>  
*nodes*)

std::shared\_ptr<*ActionInstantiation*> **actionAvailable** (*ActionInstantiation* \**instantia-*  
*tion*)

**class ActionInstantiation**

*#include <actionfactory.h>* Holds various data around action creation.

Action creation is a process, which begins with querying all actions available for certain nodes. The action factories evaluate availabilities and store some construction information here. Many parties are involved in working with it. The created action, is stored in this object as well.

## Public Functions

**ActionInstantiation** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> *select-*  
*edNodes* = {}, std::shared\_ptr<*sh::filesystem::FilesystemNode*>  
*currentDirectory* = nullptr)

Constructor for a typical fresh instance, containing infos about selected items and some more.

**ActionInstantiation** (const *ActionInstantiation* &*s*) = default

Clones an existing instance.

**ActionInstantiation** (*const ActionInstantiation* &s,  
std::shared\_ptr<sh::actions::AbstractActionItem> createdAction)  
Clones an existing instance and sets createdAction. This is a convenience constructor used for some special cases.

## Public Members

QList<std::shared\_ptr<sh::filesystem::FileSystemNode>> **selectedNodes**  
The selected nodes (initially: what's selected in the active file list).

std::shared\_ptr<sh::filesystem::FileSystemNode> **currentDirectory**  
The current directory (which is visible in the active file list).

bool **resolveLinks** = true  
If links shall be resolved for evaluation and action creation.

bool **isLookupOnCurrentDirectoryLevel** = false  
If this instantiation refers to a lookup for actions on 'current directory level' (instead of 'selection level').

*AbstractActionFactory* \***actionfactory** = 0  
The action faction.

std::shared\_ptr<ActionCategory> **category** = 0  
The action category.

std::shared\_ptr<AbstractActionItem> **createdAction** = 0  
The created action.

QKeySequence **shortcut**  
The requested keyboard shortcut for the action.

int **positionIndex** = 0  
The position information index.

**class ActionsManager** : public QObject, public *sh::base::Singleton*  
#include <actionsmanager.h> Factory for actions which are valid for a given list of FileSystemNode and tools for placing actions in some gui elements.

## Public Functions

void **getActions** (std::shared\_ptr<ActionInstantiation> *instantiation*,  
std::function<void> QList<std::shared\_ptr<ActionInstantiation>>,  
QList<std::shared\_ptr<sh::actions::ActionInstantiation>>  
> *callback*, std::function<voidstd::shared\_ptr<ActionInstantiation>> *initializedcallback*  
= [] (std::shared\_ptr< ActionInstantiation >){}) Asychonously gets actions for a list of *sh::filesystem::FileSystemNode*.

### Parameters

- *nodes*: The list of nodes.
- *callback*: This callback is called once the list is fetched.
- *initializedcallback*: This callback is called for each result action after initialization.

void **getDefaultAction** (QList<std::shared\_ptr<sh::actions::AbstractActionItem>> *action-*  
*List*, std::function<void> std::shared\_ptr<sh::actions::ActionActionItem>  
> *callback*) Determines the default action for a given list of actions.

### Parameters

- *actionList*: The list of all available actions.

- **callback**: This callback is called when the default action was found (with an already initialized action).

void **getActionForShortcut** (QList<std::shared\_ptr<filesystem::FilesystemNode>> *selectedNodes*, std::shared\_ptr<filesystem::FilesystemNode> *folderNode*, int *key*, Qt::KeyboardModifiers *modifiers*, std::function<void> std::shared\_ptr<sh::actions::AbstractActionItem> *callback*) Asychonously gets the action for a shortcut.

#### Parameters

- **selectedNodes**: The nodes which are currently selected in the active fileview.
- **folderNode**: The folder which is shown by the active fileview.
- **key**: The keypress character code.
- **modifiers**: The keypress modifiers.
- **callback**: This callback is called when an action was found (with an already initialized action).

void **registerActionFactory** (std::shared\_ptr<sh::actions::AbstractActionFactory> *actionFactory*)

Registers an action factory.

This makes an action implementation available in the toolbar and context menus.

QList<std::shared\_ptr<sh::actions::ActionItem>> **runningActions** ()

Returns a list of all currently running actions.

void **doInitialize** ()

Executes singleton initialization.

void **doShutdown** ()

Executes singleton shutdown.

void **shutdown** ()

Shutdown down this singleton.

bool **isAlive** ()

Returns if this singleton is alive (true until its shutdown begins).

## Signals

void **runningActionsChanged** ()

Emitted when the list of running actions changed.

## Public Static Functions

QString **selectionLabel** (QList<std::shared\_ptr<sh::filesystem::FilesystemNode>> *selection*)

Returns a short header label text for a node selection.

QString **directoryLabel** (std::shared\_ptr<sh::filesystem::FilesystemNode> *directory*)

Returns a short header label text for a current directory.

void **sortAndSeparateActions** (QList<std::shared\_ptr<ActionInstantiation>> *\*actions*)

Sorts a lists of actionitems and adds separators between groups. Modifies the given list.

## Private Functions

**ActionsManager** ()

QList<std::shared\_ptr<*ActionInstantiation*>> **\_getActions\_raw** (std::shared\_ptr<*ActionInstantiation*>  
instantiation)

void **\_emit\_runningActionsChanged** ()

void **recursivelyInitializeAction** (std::shared\_ptr<*sh::actions::ActionInstantiation*>  
actsit, std::function<void> std::shared\_ptr<*sh::actions::ActionInstantiation*>  
> callback, QList<std::shared\_ptr<*sh::actions::ActionInstantiation*>> sits = {} Recursively initial-  
izes a list of action items and calls a callback for each action item recursively afterwards.

### Parameters

- *actions*: List of all action items to traverse.
- *callback*: This callback is called for each action item (even indirectly in a subtree hierarchy).

void **\_getDefaultAction\_helper** (QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>>  
actionList, std::shared\_ptr<*GDAHstruct*> gdah)

Internal helper for the public *getDefaultAction()* method.

std::shared\_ptr<*sh::actions::ActionsManager::RunningActionsCounter*> **\_actionExecutionStarted** (std::shared\_ptr<*ac-*  
tion)

Helper for updating *\_runningactions*. .

## Private Members

QList<std::shared\_ptr<*sh::actions::AbstractActionFactory*>> **\_actionfactories**

QMutex **\_mutex\_actionfactories**

QList<std::shared\_ptr<*sh::actions::ActionActionItem*>> **\_runningactions**

QMutex **\_mutex\_runningactions**

## Friends

**friend class** ActionActionItem

**struct** *GDAHstruct*

Internal data structure used for *\_getDefaultAction\_helper()*;

## Public Functions

*GDAHstruct* () = default

*GDAHstruct* (const *GDAHstruct* &o) = default



## Public Members

*sh::tools::AtomicCounter* **counter**

std::function<void (std::shared\_ptr<*sh::actions::ActionActionItem*>)> **callback**

int **currentValue** = 0

QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> **initialActionList**

std::shared\_ptr<*sh::actions::AbstractActionItem*> **currentAction** = 0

**class RunningActionsCounter**

Helper for updating \_runningactions. .

## Public Functions

**RunningActionsCounter** (*ActionsManager* \**manager*,  
std::shared\_ptr<*sh::actions::ActionActionItem*> *action*)

**~RunningActionsCounter** ()

## Private Members

*ActionsManager* \*\_**manager**

std::shared\_ptr<*sh::actions::ActionActionItem*> **\_action**

**class ByRegExpPredicate : public *sh::actions::Predicate***

#include <actionfactory.h> Shows actions only when the selection paths matches a regular expression.

## Public Functions

**ByRegExpPredicate** (QString *regexp*)

bool **evaluate** (*ActionInstantiation* \**instantiation*)

Evaluates if the predicate is solved.

If it returns true, the evaluation proceeds and eventually succeeds. If false, it stops at this place.

void **prepare** (*ActionInstantiation* \**instantiation*)

Prepares the evaluation.

## Private Members

QRegExp **filterRegExp**

**class DontResolveLinksPredicate : public *sh::actions::Predicate***

#include <actionfactory.h> Disables links resolving.

## Public Functions

**DontResolveLinksPredicate** ()

void **prepare** (*ActionInstantiation* \*instantiation)  
Prepares the evaluation.

bool **evaluate** (*ActionInstantiation* \*instantiation)  
Evaluates if the predicate is solved.

If it returns `true`, the evaluation proceeds and eventually succeeds. If `false`, it stops at this place.

**class HeaderActionItem**: public *sh::actions::AbstractActionItem*  
*#include <headeractionitem.h>* A header.

Use it as all the other *AbstractActionItem* classes. It will lead to a visual header in the parent menu.

## Public Functions

**HeaderActionItem** (QString *text*, QString *icon* = QString())  
Is intended to be directly constructed from everywhere.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See *ActionDefaultPrecedenceValues* for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

```

std::weak_ptr<AbstractActionItem> parentAction ()
    Returns the parent action, if it is added to a container.

void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.

```

## Signals

```

void changed ()
    Emits when some data changed.

```

```

class HideOnCurrentDirectoryLevelPredicate : public sh::actions::Predicate
    #include <actionfactory.h> Shows actions only when not searching for 'current directory level' ac-
    tions.

```

## Public Functions

```

HideOnCurrentDirectoryLevelPredicate ()

```

```

bool evaluate (ActionInstantiation *instantiation)
    Evaluates if the predicate is solved.

```

If it returns `true`, the evaluation proceeds and eventually succeeds. If `false`, it stops at this place.

```

void prepare (ActionInstantiation *instantiation)
    Prepares the evaluation.

```

```

class HideOnSelectionLevelPredicate : public sh::actions::Predicate
    #include <actionfactory.h> Shows actions only when not searching for 'selection level' actions.

```

## Public Functions

```

HideOnSelectionLevelPredicate ()

```

```

bool evaluate (ActionInstantiation *instantiation)
    Evaluates if the predicate is solved.

```

If it returns `true`, the evaluation proceeds and eventually succeeds. If `false`, it stops at this place.

```

void prepare (ActionInstantiation *instantiation)
    Prepares the evaluation.

```

```

class KeyShortcutPredicate : public sh::actions::Predicate
    #include <actionfactory.h> Sets a keyboard shortcut.

```

### Public Functions

**KeyShortcutPredicate** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectoryLevel*)

void **prepare** (*ActionInstantiation* \**instantiation*)  
Prepares the evaluation.

bool **evaluate** (*ActionInstantiation* \**instantiation*)  
Evaluates if the predicate is solved.

If it returns `true`, the evaluation proceeds and eventually succeeds. If `false`, it stops at this place.

### Private Members

QKeySequence **shortcut**

bool **triggersOnCurrentDirectoryLevel**

**class OnDirectoriesPredicate : public *sh::actions::Predicate***  
*#include <actionfactory.h>* Shows actions only on directories.

### Public Functions

**OnDirectoriesPredicate** ()

bool **evaluate** (*ActionInstantiation* \**instantiation*)  
Evaluates if the predicate is solved.

If it returns `true`, the evaluation proceeds and eventually succeeds. If `false`, it stops at this place.

void **prepare** (*ActionInstantiation* \**instantiation*)  
Prepares the evaluation.

**class OnFilesPredicate : public *sh::actions::Predicate***  
*#include <actionfactory.h>* Shows actions only on files.

### Public Functions

**OnFilesPredicate** ()

bool **evaluate** (*ActionInstantiation* \**instantiation*)  
Evaluates if the predicate is solved.

If it returns `true`, the evaluation proceeds and eventually succeeds. If `false`, it stops at this place.

void **prepare** (*ActionInstantiation* \**instantiation*)  
Prepares the evaluation.

**class OnLinksPredicate : public *sh::actions::Predicate***  
*#include <actionfactory.h>* Shows actions only on links.

## Public Functions

**OnLinksPredicate** ()

bool **evaluate** (*ActionInstantiation* \*instantiation)

Evaluates if the predicate is solved.

If it returns `true`, the evaluation proceeds and eventually succeeds. If `false`, it stops at this place.

void **prepare** (*ActionInstantiation* \*instantiation)

Prepares the evaluation.

**class OnSingleEntrySelectionPredicate** : public *sh::actions::Predicate*  
*#include <actionfactory.h>* Shows actions only on single-entry selections.

## Public Functions

**OnSingleEntrySelectionPredicate** ()

bool **evaluate** (*ActionInstantiation* \*instantiation)

Evaluates if the predicate is solved.

If it returns `true`, the evaluation proceeds and eventually succeeds. If `false`, it stops at this place.

void **prepare** (*ActionInstantiation* \*instantiation)

Prepares the evaluation.

**class PositionIndexPredicate** : public *sh::actions::Predicate*  
*#include <actionfactory.h>* Sets a positioning information index.

## Public Functions

**PositionIndexPredicate** (int *positionIndex*)

void **prepare** (*ActionInstantiation* \*instantiation)

Prepares the evaluation.

bool **evaluate** (*ActionInstantiation* \*instantiation)

Evaluates if the predicate is solved.

If it returns `true`, the evaluation proceeds and eventually succeeds. If `false`, it stops at this place.

## Private Members

int **positionIndex**

**class Predicate**

*#include <actionfactory.h>* Controls when and how an action is created in the action factory.

Subclassed by *sh::actions::ByRegExpPredicate*, *sh::actions::DontResolveLinksPredicate*,  
*sh::actions::HideOnCurrentDirectoryLevelPredicate*, *sh::actions::HideOnSelectionLevelPredicate*,  
*sh::actions::KeyShortcutPredicate*, *sh::actions::OnDirectoriesPredicate*,  
*sh::actions::OnFilesPredicate*, *sh::actions::OnLinksPredicate*, *sh::actions::OnSingleEntrySelectionPredicate*,  
*sh::actions::PositionIndexPredicate*

## Public Functions

**Predicate** ()

void **prepare** (*ActionInstantiation* \*instantiation)

Prepares the evaluation.

bool **evaluate** (*ActionInstantiation* \*instantiation)

Evaluates if the predicate is solved.

If it returns `true`, the evaluation proceeds and eventually succeeds. If `false`, it stops at this place.

**~Predicate** ()

**class SeparatorActionItem**: public *sh::actions::AbstractActionItem*

*#include <separatoractionitem.h>* A separator.

Use it as all the other *AbstractActionItem* classes. It will lead to a visual separator in the parent menu.

## Public Functions

**SeparatorActionItem** ()

Is intended to be directly constructed from everywhere.

QString **text** ()

Returns the displayed text for this action.

QString **icon** ()

Returns the icon for this action.

bool **enabled** ()

Checks if this action is enabled.

bool **isChecked** ()

Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See *ActionDefaultPrecedenceValues* for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

```

bool visible ()
    Checks the visibility of this item (non-recursively).

std::weak_ptr<AbstractActionItem> parentAction ()
    Returns the parent action, if it is added to a container.

void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action.

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.

```

## Signals

```

void changed ()
    Emits when some data changed.

```

```

class SubmenuItemActionItem: public sh::actions::AbstractActionItem
    #include <submenuactionitem.h> Abstract base class for a submenu.

```

It can be made visible in menus or the toolbar or executed directly.

Subclasses fill the submenu with other actions in order to be useful.

Subclassed by *sh::actions::common::ActionAbstractTransferTo*, *sh::actions::common::ActionBookmarkFolder*, *sh::actions::common::ActionGroups*, *sh::actions::common::ActionHistoryNavigate*, *sh::actions::common::ActionOpenFileWith*, *sh::actions::common::ActionOpenTerminal*, *sh::actions::mainmenu::ActionApplyProfile*, *sh::actions::mainmenu::ActionMain*, *sh::actions::mainmenu::ActionNumberFileviews*, *sh::scripting::api::ApiSubmenuItemActionItem*

## Public Functions

```

SubmenuItemActionItem (QString text, const QList<std::shared_ptr<sh::actions::AbstractActionItem>>
    subitems = {}, bool enabled = true, QString icon = QString(), int
    defaultActionPrecedence = 0, bool checkable = false, bool ischecked
    = false)
    Is (for subclasses) intended to be directly constructed from everywhere or by registering a factory
    somewhere.

```

```

const QList<std::shared_ptr<sh::actions::AbstractActionItem>> subitems ()
    Returns the list of subitems from this submenu.

```

```

void setSubitems (const      QList<std::shared_ptr<sh::actions::AbstractActionItem>>
                  subitems)
    Sets the subitems.

```

```

QString text ()
    Returns the displayed text for this action.

```

```

QString icon ()
    Returns the icon for this action.

```

**bool enabled ()**  
Checks if this action is enabled.

**bool isChecked ()**  
Checks if this action is checked (has a cross in the ui).

**bool isCheckable ()**  
Checks if this action is checkable (can have a cross in the ui).

**int defaultActionPrecedence () const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

**void setText (QString text)**  
Sets the displayed text.

**void setIcon (QString icon)**  
Sets the icon.

**void setEnabled (bool enabled)**  
Sets if the item is enabled.

**void setChecked (bool checked)**  
Sets if the item is checked (has a cross in the ui).

**void setVisible (bool visible)**  
Sets the visibility of this item.

**bool visible ()**  
Checks the visibility of this item (non-recursively).

**std::weak\_ptr<AbstractActionItem> parentAction ()**  
Returns the parent action, if it is added to a container.

**void \_setParentAction (std::shared\_ptr<AbstractActionItem> parent)**  
Sets the parent action. .

**void initializeAsync (std::function<void>  
> oninitialized = 0)**Asynchronously initializes the action.

**void initializeSync ()**  
Synchronously initializes the action.

**bool isInitialized ()**  
Checks if this action is initialized.

**bool isInitializing ()**  
Checks if this action is initializing.



## Signals

void **subitemsChanged** ()  
 Emits when the list of subitems changed.

void **changed** ()  
 Emits when some data changed.

## Private Members

QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> **\_subitems**

### namespace common

Implementations of actions.

Subclasses of *sh::actions::AbstractActionItem* (and possibly some auxiliary stuff). Many action implementations used in Shallot are here. You can find other actions in the sibling namespaces and a few very special ones at different places.

**class ActionAbstractTransferTo : public *sh::actions::SubmenuItem***  
*#include <actiontransfer.h>* Abstract action for transferring a selection to some other locations (e.g. to other file views, bookmarks).  
 Subclassed by *sh::actions::common::ActionTransferToCopy*,  
*sh::actions::common::ActionTransferToMove*

## Public Functions

**ActionAbstractTransferTo** (QString *text*, QString *icon*,  
 QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>>  
*nodes*)

**const** QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> **subitems** ()  
 Returns the list of subitems from this submenu.

void **setSubitems** (**const** QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>>  
*subitems*)  
 Sets the subitems.

QString **text** ()  
 Returns the displayed text for this action.

QString **icon** ()  
 Returns the icon for this action.

bool **enabled** ()  
 Checks if this action is enabled.

bool **isChecked** ()  
 Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
 Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
 Returns the precedence for becoming the default action of a parent submenu.  
 This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See *ActionDefaultPrecedenceValues* for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<AbstractActionItem> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<AbstractActionItem> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0)  
Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **subitemsChanged** ()  
Emits when the list of subitems changed.

void **changed** ()  
Emits when some data changed.

## Friends

**friend class** ActionTransferToHelper

**class** ActionAbstractTransferTree : public *sh::actions::ActionActionItem*  
*#include <actionabstracttransfertree.h>* Abstract action for transferring (copying, moving, et al)  
a tree of items.

Subclassed by *sh::actions::common::ActionCopyTree*

## Public Functions

**ActionAbstractTransferTree** (QList<std::shared\_ptr<const *sh::filesystem::Eurl*>>  
sources, std::shared\_ptr<const *sh::filesystem::Eurl*>  
destination)

**~ActionAbstractTransferTree** ()

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \*info)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence shortcut, bool triggersOnCurrentDirectory = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () const  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString text)  
Sets the displayed text.

void **setIcon** (QString icon)  
Sets the icon.

void **setEnabled** (bool enabled)  
Sets if the item is enabled.

void **setChecked** (bool checked)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool visible)  
Sets the visibility of this item.

`bool visible ()`  
Checks the visibility of this item (non-recursively).

`std::weak_ptr<AbstractActionItem> parentAction ()`  
Returns the parent action, if it is added to a container.

`void _setParentAction (std::shared_ptr<AbstractActionItem> parent)`  
Sets the parent action. .

`void initializeAsync (std::function<void>  
> oninitialized = 0)`  
Asynchronously initializes the action.

`void initializeSync ()`  
Synchronously initializes the action.

`bool isInitialized ()`  
Checks if this action is initialized.

`bool isInitializing ()`  
Checks if this action is initializing.

## Signals

`void changed ()`  
Emits when some data changed.

## Public Static Functions

`void makereadableunits (double *value, QString *vunit)`

`class MyFilesystemOperationProgressMonitor : public sh::filesystem::FilesystemOperationProg  
#include <actionabstracttransfertree.h>`

## Public Functions

`MyFilesystemOperationProgressMonitor (ActionExecutionInfo *info)`

`bool hasItemInfo ()`  
Checks if this progress monitor provides information about how many items of a certain total number are transferred so far.

`quint64 doneItems ()`  
Checks how many items are transferred so far.

`quint64 allItems ()`  
Checks how many items are to be transferred in total (as predicted in the current moment).

`bool hasBytesInfo ()`  
Checks if this progress monitor provides information about how many byte of a certain total number are transferred so far.

`quint64 doneBytes ()`  
Checks how many bytes are transferred so far.

`quint64 allBytes ()`  
Checks how many bytes are to be transferred in total (as predicted in the current moment).

`QString getItemInfoFrom ()`  
Returns the current source of transfer (as textual information).

QString **getItemInfoTo** ()

Returns the current destination of transfer (as textual information).

QString **estimation** ()

Returns the current performance and time estimation (as textual information).

**class ActionAddToBookmarks** : public *sh::actions::ActionActionItem*

*#include <actionbookmarks.h>* Action for bookmarking a directory.

## Public Functions

**ActionAddToBookmarks** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>>  
nodes)

void **execute** ()

Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \*info)

Executes this action.

QKeySequence **shortcuthint** ()

Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()

Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence shortcut, bool triggersOnCurrentDirectory =  
false)

Sets the keyboard shortcut for triggering this action.

QString **text** ()

Returns the displayed text for this action.

QString **icon** ()

Returns the icon for this action.

bool **enabled** ()

Checks if this action is enabled.

bool **isChecked** ()

Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () const

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See *ActionDefaultPrecedenceValues* for reference values.

void **setText** (QString text)

Sets the displayed text.

void **setIcon** (QString icon)

Sets the icon.

void **setEnabled** (bool enabled)

Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<*AbstractActionItem*> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<*AbstractActionItem*> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

## Public Static Functions

void **doInitialize** ()  
void **doShutdown** ()

**class ActionBookmark** : public *sh::actions::ActionActionItem*  
*#include <actionbookmarks.h>* Action for navigating to a certain bookmark.

## Public Functions

**ActionBookmark** (QString *label*, std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*)

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcutHint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

**class ActionBookmarkFolder** : public *sh::actions::SubmenuItem*  
*#include <actionbookmarks.h>* Submenu action of bookmarks.  
Subclassed by *sh::actions::common::ActionBookmarks*

## Public Functions

**ActionBookmarkFolder** (QString *label*)

void **addSubitem** (std::shared\_ptr<*sh::actions::AbstractActionItem*> *item*)

void **clearSubitems** ()

**const** QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> **subitems** ()  
Returns the list of subitems from this submenu.

void **setSubitems** (**const**       QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>>  
                                  *subitems*)  
Sets the subitems.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [\*ActionDefaultPrecedenceValues\*](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.



```

bool visible ()
    Checks the visibility of this item (non-recursively).

std::weak_ptr<AbstractActionItem> parentAction ()
    Returns the parent action, if it is added to a container.

void __setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.

```

## Signals

```

void subitemsChanged ()
    Emits when the list of subitems changed.

void changed ()
    Emits when some data changed.

```

```

class ActionBookmarks : public sh::actions::common::ActionBookmarkFolder
    #include <actionbookmarks.h> Bookmarking action (the main one you see in the toolbar)

```

## Public Functions

```

ActionBookmarks ()

void initialize ()
    Initialize the action. This should make the time-consuming parts, e.g. for determining a label
    or enabled state.

void addSubitem (std::shared_ptr<sh::actions::AbstractActionItem> item)

void clearSubitems ()

const QList<std::shared_ptr<sh::actions::AbstractActionItem>> subitems ()
    Returns the list of subitems from this submenu.

void setSubitems (const      QList<std::shared_ptr<sh::actions::AbstractActionItem>>
                  subitems)
    Sets the subitems.

QString text ()
    Returns the displayed text for this action.

QString icon ()
    Returns the icon for this action.

bool enabled ()
    Checks if this action is enabled.

```

bool **isChecked** ()

Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>)

> *oninitialized* = 0 Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **subitemsChanged** ()

Emits when the list of subitems changed.

void **changed** ()

Emits when some data changed.

**class ActionClipboardCopy : public [sh::actions::ActionActionItem](#)**

**#include <actionclipboard.h>** Action for copying to clipboard.

## Public Functions

**ActionClipboardCopy** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>>  
nodes)

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \*info)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence shortcut, bool triggersOnCurrentDirectory =  
false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString text)  
Sets the displayed text.

void **setIcon** (QString icon)  
Sets the icon.

void **setEnabled** (bool enabled)  
Sets if the item is enabled.

void **setChecked** (bool checked)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool visible)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

`std::weak_ptr<AbstractActionItem> parentAction ()`  
Returns the parent action, if it is added to a container.

`void _setParentAction (std::shared_ptr<AbstractActionItem> parent)`  
Sets the parent action. .

`void initializeAsync (std::function<void>  
> oninitialized = 0)`Asynchronously initializes the action.

`void initializeSync ()`  
Synchronously initializes the action.

`bool isInitialized ()`  
Checks if this action is initialized.

`bool isInitializing ()`  
Checks if this action is initializing.

### Signals

`void changed ()`  
Emits when some data changed.

### Public Static Functions

`void doInitialize ()`

`void doShutdown ()`

**class ActionClipboardCut : public *sh::actions::ActionActionItem***  
*#include <actionclipboard.h>* Action for cutting to clipboard.

### Public Functions

**ActionClipboardCut** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> nodes)

`void execute ()`  
Executes this action.

`void execute (sh::actions::ActionExecutionInfo *info)`  
Executes this action.

`QKeySequence shortcuthint ()`  
Returns the keyboard shortcut for triggering this action.

`bool shortcuthintTriggersOnCurrentDirectory ()`  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

`void setShortcuthint (QKeySequence shortcut, bool triggersOnCurrentDirectory = false)`  
Sets the keyboard shortcut for triggering this action.

`QString text ()`  
Returns the displayed text for this action.

`QString icon ()`  
Returns the icon for this action.

bool **enabled** ()  
 Checks if this action is enabled.

bool **isChecked** ()  
 Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
 Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
 Returns the precedence for becoming the default action of a parent submenu.  
 This e.g. leads to a bold label.  
 The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
 Sets the displayed text.

void **setIcon** (QString *icon*)  
 Sets the icon.

void **setEnabled** (bool *enabled*)  
 Sets if the item is enabled.

void **setChecked** (bool *checked*)  
 Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
 Sets the visibility of this item.

bool **visible** ()  
 Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
 Returns the parent action, if it is added to a container.

void **\_\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
 Sets the parent action. .

void **initializeAsync** (std::function<void>  
 > *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
 Synchronously initializes the action.

bool **isInitialized** ()  
 Checks if this action is initialized.

bool **isInitializing** ()  
 Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

## Public Static Functions

void **doInitialize** ()

void **doShutdown** ()

**class ActionClipboardPaste** : public *sh::actions::ActionActionItem*  
*#include <actionclipboard.h>* Action for pasting from clipboard.

## Public Functions

**ActionClipboardPaste** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>>  
*nodes*)

bool **shortcutTriggersOnFolder** ()

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \*info)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* =  
false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

```

void setText (QString text)
    Sets the displayed text.

void setIcon (QString icon)
    Sets the icon.

void setEnabled (bool enabled)
    Sets if the item is enabled.

void setChecked (bool checked)
    Sets if the item is checked (has a cross in the ui).

void setVisible (bool visible)
    Sets the visibility of this item.

bool visible ()
    Checks the visibility of this item (non-recursively).

std::weak_ptr<AbstractActionItem> parentAction ()
    Returns the parent action, if it is added to a container.

void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.

```

## Signals

```

void changed ()
    Emits when some data changed.

```

## Public Static Functions

```

void doInitialize ()

void doShutdown ()

```

```

class ActionClipboardPasteAsLink : public sh::actions::ActionActionItem
    #include <actionclipboard.h> Action for pasting from clipboard as link.

```

## Public Functions

**ActionClipboardPasteAsLink** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>>  
nodes)

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \*info)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence shortcut, bool triggersOnCurrentDirectory =  
false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString text)  
Sets the displayed text.

void **setIcon** (QString icon)  
Sets the icon.

void **setEnabled** (bool enabled)  
Sets if the item is enabled.

void **setChecked** (bool checked)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool visible)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).



```
std::weak_ptr<AbstractActionItem> parentAction ()
    Returns the parent action, if it is added to a container.

void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.
```

## Signals

```
void changed ()
    Emits when some data changed.
```

## Public Static Functions

```
void doInitialize ()
void doShutdown ()
```

```
class ActionCopyTree : public sh::actions::common::ActionAbstractTransferTree
    #include <actioncopytree.h> Action copying a filesystem tree to some other place.

    Subclassed by sh::actions::common::ActionMoveTree
```

## Public Functions

```
ActionCopyTree (QList<std::shared_ptr<const sh::filesystem::Eurl>> sources,
    std::shared_ptr<const sh::filesystem::Eurl> destination)

void action (sh::actions::ActionExecutionInfo *info)
    The action implementation, i.e. what the actions should actually do.

void execute ()
    Executes this action.

void execute (sh::actions::ActionExecutionInfo *info)
    Executes this action.

QKeySequence shortcuthint ()
    Returns the keyboard shortcut for triggering this action.

bool shortcuthintTriggersOnCurrentDirectory ()
    Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the
    entry selection).

void setShortcuthint (QKeySequence shortcut, bool triggersOnCurrentDirectory =
    false)
    Sets the keyboard shortcut for triggering this action.
```

QString **text** ()

Returns the displayed text for this action.

QString **icon** ()

Returns the icon for this action.

bool **enabled** ()

Checks if this action is enabled.

bool **isChecked** ()

Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>

> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

## Public Static Functions

void **makereadableunits** (double *\*value*, QString *\*vunit*)

**class ActionCreateFile** : public *sh::actions::ActionActionItem*  
*#include <actioncreatefile.h>* Action for creating a new file.

## Public Functions

**ActionCreateFile** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> *nodes*)

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* *\*info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

```
void setEnabled (bool enabled)  
    Sets if the item is enabled.  
  
void setChecked (bool checked)  
    Sets if the item is checked (has a cross in the ui).  
  
void setVisible (bool visible)  
    Sets the visibility of this item.  
  
bool visible ()  
    Checks the visibility of this item (non-recursively).  
  
std::weak_ptr<AbstractActionItem> parentAction ()  
    Returns the parent action, if it is added to a container.  
  
void _setParentAction (std::shared_ptr<AbstractActionItem> parent)  
    Sets the parent action. .  
  
void initializeAsync (std::function<void>  
    > oninitialized = 0) Asynchronously initializes the action.  
  
void initializeSync ()  
    Synchronously initializes the action.  
  
bool isInitialized ()  
    Checks if this action is initialized.  
  
bool isInitializing ()  
    Checks if this action is initializing.
```

## Signals

```
void changed ()  
    Emits when some data changed.
```

## Public Static Functions

```
void doInitialize ()  
  
void doShutdown ()
```

```
class ActionCreateFolder : public sh::actions::ActionActionItem  
    #include <actioncreatefolder.h> Action for creating a new directory.
```

## Public Functions

```
ActionCreateFolder (QList<std::shared_ptr<sh::filesystem::FilesystemNode>> nodes)  
  
bool shortcutTriggersOnFolder ()  
  
void execute ()  
    Executes this action.  
  
void execute (sh::actions::ActionExecutionInfo *info)  
    Executes this action.  
  
QKeySequence shortcuthint ()  
    Returns the keyboard shortcut for triggering this action.
```

bool **shortcutHintTriggersOnCurrentDirectory** ()  
 Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcutHint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
 Sets the keyboard shortcut for triggering this action.

QString **text** ()  
 Returns the displayed text for this action.

QString **icon** ()  
 Returns the icon for this action.

bool **enabled** ()  
 Checks if this action is enabled.

bool **isChecked** ()  
 Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
 Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () const  
 Returns the precedence for becoming the default action of a parent submenu.  
 This e.g. leads to a bold label.  
 The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
 Sets the displayed text.

void **setIcon** (QString *icon*)  
 Sets the icon.

void **setEnabled** (bool *enabled*)  
 Sets if the item is enabled.

void **setChecked** (bool *checked*)  
 Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
 Sets the visibility of this item.

bool **visible** ()  
 Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
 Returns the parent action, if it is added to a container.

void **\_\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
 Sets the parent action. .

void **initializeAsync** (std::function<void> > *oninitialized* = 0)  
 Asynchronously initializes the action.

void **initializeSync** ()  
 Synchronously initializes the action.

bool **isInitialized** ()  
 Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

## Public Static Functions

void **doInitialize** ()

void **doShutdown** ()

**class ActionDeleteItems** : public *sh::actions::ActionActionItem*  
*#include <actiondeleteitems.h>* Action deleting a list of filesystem items.

## Public Functions

**ActionDeleteItems** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> *nodes*)

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See *ActionDefaultPrecedenceValues* for reference values.

```

void setText (QString text)
    Sets the displayed text.

void setIcon (QString icon)
    Sets the icon.

void setEnabled (bool enabled)
    Sets if the item is enabled.

void setChecked (bool checked)
    Sets if the item is checked (has a cross in the ui).

void setVisible (bool visible)
    Sets the visibility of this item.

bool visible ()
    Checks the visibility of this item (non-recursively).

std::weak_ptr<AbstractActionItem> parentAction ()
    Returns the parent action, if it is added to a container.

void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.

```

## Signals

```

void changed ()
    Emits when some data changed.

```

## Public Static Functions

```

void doInitialize ()

void doShutdown ()

```

```

class ActionExecute : public sh::actions::ActionActionItem
    #include <actionexecute.h> Action executing a file (if executable).

```

## Public Functions

**ActionExecute** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> *nodes*)

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<*AbstractActionItem*> **parentAction** ()  
Returns the parent action, if it is added to a container.



```

void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.

```

## Signals

```

void changed ()
    Emits when some data changed.

```

## Public Static Functions

```

void doInitialize ()

void doShutdown ()

```

```

class ActionGroups : public sh::actions::SubmenuItem
    #include <actiongroups.h> Action for showing registered actions grouped by an
    sh::actions::ActionCategory.

```

## Public Functions

```

ActionGroups (std::shared_ptr<sh::actions::ActionCategory> category)

void setGroupActions (std::shared_ptr<sh::actions::ActionInstantiation> rootsituation,
    QList<std::shared_ptr<sh::actions::ActionInstantiation>>
    selects, QList<std::shared_ptr<sh::actions::ActionInstantiation>>
    directs)

const QList<std::shared_ptr<sh::actions::AbstractActionItem>> subitems ()
    Returns the list of subitems from this submenu.

void setSubitems (const      QList<std::shared_ptr<sh::actions::AbstractActionItem>>
    subitems)
    Sets the subitems.

QString text ()
    Returns the displayed text for this action.

QString icon ()
    Returns the icon for this action.

bool enabled ()
    Checks if this action is enabled.

bool isChecked ()
    Checks if this action is checked (has a cross in the ui).

```

bool **isCheckedable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>

> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **subitemsChanged** ()

Emits when the list of subitems changed.

void **changed** ()

Emits when some data changed.

**class ActionHistoryNavigate**: public [sh::actions::SubmenuItem](#)

*#include <actionnavigation.h>* Abstract action for navigating in the directory history stack (step-wise or randomly).

Subclassed by [sh::actions::common::ActionHistoryNavigateBackward](#),  
[sh::actions::common::ActionHistoryNavigateForward](#)

## Public Functions

**ActionHistoryNavigate** (QString *text*, std::function<QList<[sh::tools::HistoryTracker](#)<std::shared\_ptr<con.  
[sh::filesystem::Eurl](#)>>::HistoryEntry>)  
 > *\_getentries* fct QString *icon*

**const** QList<std::shared\_ptr<[sh::actions::AbstractActionItem](#)>> **subitems** ()  
 Returns the list of subitems from this submenu.

void **setSubitems** (**const** QList<std::shared\_ptr<[sh::actions::AbstractActionItem](#)>>  
*subitems*)  
 Sets the subitems.

QString **text** ()  
 Returns the displayed text for this action.

QString **icon** ()  
 Returns the icon for this action.

bool **enabled** ()  
 Checks if this action is enabled.

bool **isChecked** ()  
 Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
 Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
 Returns the precedence for becoming the default action of a parent submenu.  
 This e.g. leads to a bold label.  
 The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
 Sets the displayed text.

void **setIcon** (QString *icon*)  
 Sets the icon.

void **setEnabled** (bool *enabled*)  
 Sets if the item is enabled.

void **setChecked** (bool *checked*)  
 Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
 Sets the visibility of this item.

bool **visible** ()  
 Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
 Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
 Sets the parent action. .

void **initializeAsync** (std::function<void>  
 > *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **subitemsChanged** ()  
Emits when the list of subitems changed.

void **changed** ()  
Emits when some data changed.

**class ActionHistoryNavigateBackward**: public *sh::actions::common::ActionHistoryNavigate*  
*#include <actionnavigation.h>* Action for navigating backward in the directory history stack.

## Public Functions

**ActionHistoryNavigateBackward** ()

const QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> **subitems** ()  
Returns the list of subitems from this submenu.

void **setSubitems** (const QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>>  
subitems)  
Sets the subitems.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () const  
Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString text)  
Sets the displayed text.

void **setIcon** (QString icon)  
Sets the icon.

```

void setEnabled (bool enabled)
    Sets if the item is enabled.

void setChecked (bool checked)
    Sets if the item is checked (has a cross in the ui).

void setVisible (bool visible)
    Sets the visibility of this item.

bool visible ()
    Checks the visibility of this item (non-recursively).

std::weak_ptr<AbstractActionItem> parentAction ()
    Returns the parent action, if it is added to a container.

void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.

```

## Signals

```

void subitemsChanged ()
    Emits when the list of subitems changed.

void changed ()
    Emits when some data changed.

```

```

class ActionHistoryNavigateForward: public sh::actions::common::ActionHistoryNavigate
    #include <actionnavigation.h> Action for navigating forward in the directory history stack.

```

## Public Functions

```

ActionHistoryNavigateForward ()

const QList<std::shared_ptr<sh::actions::AbstractActionItem>> subitems ()
    Returns the list of subitems from this submenu.

void setSubitems (const      QList<std::shared_ptr<sh::actions::AbstractActionItem>>
                  subitems)
    Sets the subitems.

QString text ()
    Returns the displayed text for this action.

QString icon ()
    Returns the icon for this action.

bool enabled ()
    Checks if this action is enabled.

```

bool **isChecked** ()

Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>)

> *oninitialized* = 0 Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **subitemsChanged** ()

Emits when the list of subitems changed.

void **changed** ()

Emits when some data changed.

**class ActionManageBookmarks : public [sh::actions::ActionActionItem](#)**

**#include <actionbookmarks.h>** Action for managing bookmarks.

## Public Functions

### ActionManageBookmarks ( )

void **execute** ( )  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo \*info*)  
Executes this action.

QKeySequence **shortcuthint** ( )  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ( )  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ( )  
Returns the displayed text for this action.

QString **icon** ( )  
Returns the icon for this action.

bool **enabled** ( )  
Checks if this action is enabled.

bool **isChecked** ( )  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ( )  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** ( ) **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ( )  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ( )  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<*AbstractActionItem*> parent)

Sets the parent action. .

void **initializeAsync** (std::function<void>)

> *oninitialized* = 0 Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **changed** ()

Emits when some data changed.

**class ActionMoveTree : public *sh::actions::common::ActionCopyTree***

*#include <actionmovetree.h>* Action moving a filesystem tree to some other place.

## Public Functions

**ActionMoveTree** (QList<std::shared\_ptr<const *sh::filesystem::Eurl*>> *sources*,  
std::shared\_ptr<const *sh::filesystem::Eurl*> *destination*)

void **action** (*sh::actions::ActionExecutionInfo* \**info*)

The action implementation, i.e. what the actions should actually do.

void **execute** ()

Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)

Executes this action.

QKeySequence **shortcuthint** ()

Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()

Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* =  
false)

Sets the keyboard shortcut for triggering this action.

QString **text** ()

Returns the displayed text for this action.

QString **icon** ()

Returns the icon for this action.

bool **enabled** ()

Checks if this action is enabled.

bool **isChecked** ()

Checks if this action is checked (has a cross in the ui).



bool **isCheckedable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>

> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **changed** ()

Emits when some data changed.

## Public Static Functions

void **makereadableunits** (double *\*value*, QString *\*vunit*)

**class ActionNavigateInHistory** : public *sh::actions::ActionActionItem*  
*#include <actionnavigation.h>* Action for navigating to one particular place in the directory history stack.

## Public Functions

**ActionNavigateInHistory** (*sh::ui::FileView* *\*filelist*, int *idx*, std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*, bool *isdefault*)

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* *\*info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.  
The action with the highest value becomes the default. Only values >0 will be considered. See *ActionDefaultPrecedenceValues* for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

```

void setChecked (bool checked)
    Sets if the item is checked (has a cross in the ui).

void setVisible (bool visible)
    Sets the visibility of this item.

bool visible ()
    Checks the visibility of this item (non-recursively).

std::weak_ptr<AbstractActionItem> parentAction ()
    Returns the parent action, if it is added to a container.

void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.

```

## Signals

```

void changed ()
    Emits when some data changed.

```

```

class ActionOpenArchive : public sh::actions::ActionActionItem
    #include <actionopenarchive.h> Action opening a share archive (by creating a root node for it).

    Subclassed by sh::filesystemhandlers::ArchiveFilesystemHandler::ActionOpenTarArchive,
    sh::filesystemhandlers::ArchiveFilesystemHandler::ActionOpenZipArchive

```

## Public Functions

```

ActionOpenArchive (QList<std::shared_ptr<sh::filesystem::FilesystemNode>> nodes)

void execute ()
    Executes this action.

void execute (sh::actions::ActionExecutionInfo *info)
    Executes this action.

QKeySequence shortcuthint ()
    Returns the keyboard shortcut for triggering this action.

bool shortcuthintTriggersOnCurrentDirectory ()
    Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the
    entry selection).

void setShortcuthint (QKeySequence shortcut, bool triggersOnCurrentDirectory =
    false)
    Sets the keyboard shortcut for triggering this action.

QString text ()
    Returns the displayed text for this action.

```

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

**class ActionOpenDirectoryInNewWindow** : public *sh::actions::ActionActionItem*  
*#include <actionopendirectoryinnewwindow.h>* Action for opening a file with a automatically  
chosen program.

## Public Functions

**ActionOpenDirectoryInNewWindow** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>>  
*nodes*)

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the  
entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* =  
false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.  
The action with the highest value becomes the default. Only values >0 will be considered. See  
[ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<*AbstractActionItem*> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_\_setParentAction** (std::shared\_ptr<*AbstractActionItem*> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

## Public Static Functions

void **doInitialize** ()  
void **doShutdown** ()

**class ActionOpenFile** : public *sh::actions::ActionActionItem*  
*#include <actionopenfile.h>* Action for opening a file with a automatically chosen program.

## Public Functions

**ActionOpenFile** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> *nodes*)

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcutHint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
 Sets the keyboard shortcut for triggering this action.

QString **text** ()  
 Returns the displayed text for this action.

QString **icon** ()  
 Returns the icon for this action.

bool **enabled** ()  
 Checks if this action is enabled.

bool **isChecked** ()  
 Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
 Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
 Returns the precedence for becoming the default action of a parent submenu.  
 This e.g. leads to a bold label.  
 The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
 Sets the displayed text.

void **setIcon** (QString *icon*)  
 Sets the icon.

void **setEnabled** (bool *enabled*)  
 Sets if the item is enabled.

void **setChecked** (bool *checked*)  
 Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
 Sets the visibility of this item.

bool **visible** ()  
 Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
 Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
 Sets the parent action. .

void **initializeAsync** (std::function<void>  
 > *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
 Synchronously initializes the action.

bool **isInitialized** ()  
 Checks if this action is initialized.

bool **isInitializing** ()  
 Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

## Public Static Functions

void **doInitialize** ()

void **doShutdown** ()

**class ActionOpenFileWith**: public *sh::actions::SubmenuItem*  
*#include <actionopenfilewith.h>* Submenu action for opening a file with a manually chosen program.

## Public Functions

**ActionOpenFileWith** (QList<std::shared\_ptr<*sh::filesystem::FileSystemNode*>> *nodes*)

**const** QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> **subitems** ()  
Returns the list of subitems from this submenu.

void **setSubitems** (**const**      QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>>  
                                 *subitems*)  
Sets the subitems.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.  
The action with the highest value becomes the default. Only values >0 will be considered. See [\*ActionDefaultPrecedenceValues\*](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).



```

void setVisible (bool visible)
    Sets the visibility of this item.

bool visible ()
    Checks the visibility of this item (non-recursively).

std::weak_ptr<AbstractActionItem> parentAction ()
    Returns the parent action, if it is added to a container.

void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.

```

## Signals

```

void subitemsChanged ()
    Emits when the list of subitems changed.

void changed ()
    Emits when some data changed.

```

## Public Static Functions

```

void waitProgramClosedIfNeeded (sh::actions::ActionExecutionInfo *info,
                                QStringList filelist)

void doInitialize ()

void doShutdown ()

```

## Friends

```

friend class ActionOpenFile

class _ActionOpenFileWithEntry : public sh::actions::ActionActionItem

```

## Public Functions

```

_ActionOpenFileWithEntry (QList<std::shared_ptr<sh::filesystem::FilesystemNode>>
                           nodes, QString name, QString cmd, QStringList argu-
                           ments, QStringList rememberformimetype)

void action (sh::actions::ActionExecutionInfo *info)
    The action implementation, i.e. what the actions should actually do.

void execute ()
    Executes this action.

```

void **execute** (*sh::actions::ActionExecutionInfo \*info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered.  
See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>)  
> *oninitialized* = 0Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Public Members

QString **\_command**

QStringList **\_commandArguments**

QList<std::shared\_ptr<sh::filesystem::FilesystemNode>> **\_nodes**

## Signals

void **changed** ()  
Emits when some data changed.

**class \_ActionOpenFileWithUI : public sh::actions::ActionActionItem**

## Public Functions

**\_ActionOpenFileWithUI** (QList<std::shared\_ptr<sh::filesystem::FilesystemNode>>  
*nodes*, QString *mimetype*)

void **action** (sh::actions::ActionExecutionInfo \**info*)  
The action implementation, i.e. what the actions should actually do.

void **execute** ()  
Executes this action.

void **execute** (sh::actions::ActionExecutionInfo \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* =  
false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () const

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered.

See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>)

> *oninitialized* = 0 Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Public Members

QList<std::shared\_ptr<[sh::filesystem::FilesystemNode](#)>> **\_nodes**

QString **\_mimetype**

## Signals

void **changed** ()  
Emits when some data changed.

**class ActionOpenSharcArchive : public *sh::actions::ActionActionItem***  
*#include <actionopensharcarchive.h>* Action opening a sharc archive (by creating a root node for it).

## Public Functions

**ActionOpenSharcArchive** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>>  
*nodes*)

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* =  
false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.  
The action with the highest value becomes the default. Only values >0 will be considered. See [\*ActionDefaultPrecedenceValues\*](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

```
void setChecked(bool checked)
    Sets if the item is checked (has a cross in the ui).

void setVisible(bool visible)
    Sets the visibility of this item.

bool visible()
    Checks the visibility of this item (non-recursively).

std::weak_ptr<AbstractActionItem> parentAction()
    Returns the parent action, if it is added to a container.

void _setParentAction(std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action.

void initializeAsync(std::function<void>
    > oninitialized) Asynchronously initializes the action.

void initializeSync()
    Synchronously initializes the action.

bool isInitialized()
    Checks if this action is initialized.

bool isInitializing()
    Checks if this action is initializing.
```

## Signals

```
void changed ()
```

Emits when some data changed.

## Public Static Functions

```
void doInitialize ()
void doShutdown ()
```

```
class ActionOpenTerminal : public sh::actions::SubmenuItem
#include <actionopenterminal.h> Action submenu with actions for opening commandline terminals.
```

## Public Functions

**ActionOpenTerminal** (QString *dir*)

```
void initialize()
    Initialize the action. This should make the time-consuming parts, e.g. for determining a label
    or enabled state.
```

**const** QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> **subitems** ()  
Returns the list of subitems from this submenu.

<pre>void <b>setSubitems</b> (<b>const</b>                   <i>subitems</i>)</pre>	<pre>QList&lt;std::shared_ptr&lt;<i>sh::actions::AbstractActionItem</i>&gt;&gt;</pre>
<p>Sets the subitems.</p>	

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **subitemsChanged** ()  
Emits when the list of subitems changed.

void **changed** ()  
Emits when some data changed.

**class ActionOpenTerminalAsRoot : public *sh::actions::common::ActionOpenTerminalAsUser***  
*#include <actionopenterminal.h>* Action for opening a commandline terminal as root.

## Public Functions

**ActionOpenTerminalAsRoot** (QString *dir*)

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [\*ActionDefaultPrecedenceValues\*](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.



```

void setEnabled (bool enabled)
    Sets if the item is enabled.

void setChecked (bool checked)
    Sets if the item is checked (has a cross in the ui).

void setVisible (bool visible)
    Sets the visibility of this item.

bool visible ()
    Checks the visibility of this item (non-recursively).

std::weak_ptr<AbstractActionItem> parentAction ()
    Returns the parent action, if it is added to a container.

void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.

```

## Signals

```

void changed ()
    Emits when some data changed.

```

```

class ActionOpenTerminalAsUser : public sh::actions::ActionActionItem
    #include <actionopenterminal.h> Action for opening a commandline terminal (with current
    user).

    Subclassed by sh::actions::common::ActionOpenTerminalAsRoot

```

## Public Functions

```

ActionOpenTerminalAsUser (QString dir)

void execute ()
    Executes this action.

void execute (sh::actions::ActionExecutionInfo *info)
    Executes this action.

QKeySequence shortcuthint ()
    Returns the keyboard shortcut for triggering this action.

bool shortcuthintTriggersOnCurrentDirectory ()
    Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the
    entry selection).

void setShortcuthint (QKeySequence shortcut, bool triggersOnCurrentDirectory =
    false)
    Sets the keyboard shortcut for triggering this action.

```

QString **text** ()

Returns the displayed text for this action.

QString **icon** ()

Returns the icon for this action.

bool **enabled** ()

Checks if this action is enabled.

bool **isChecked** ()

Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>

> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

**class ActionRenameItem**: public *sh::actions::ActionActionItem*  
*#include <actionrenameitem.h>* Action for renaming items.

## Public Functions

**ActionRenameItem** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> *nodes*)

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0)  
Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

## Public Static Functions

void **doInitialize** ()  
void **doShutdown** ()

**class ActionShowProperties** : public [sh::actions::ActionActionItem](#)  
*#include <actionshowproperties.h>* Action for showing the properties dialog for filesystem nodes.

## Public Functions

**ActionShowProperties** (QList<std::shared\_ptr<[sh::filesystem::FilesystemNode](#)>>  
*nodes*)

void **execute** ()  
Executes this action.

void **execute** ([sh::actions::ActionExecutionInfo](#) \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcutHint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
 Sets the keyboard shortcut for triggering this action.

QString **text** ()  
 Returns the displayed text for this action.

QString **icon** ()  
 Returns the icon for this action.

bool **enabled** ()  
 Checks if this action is enabled.

bool **isChecked** ()  
 Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
 Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
 Returns the precedence for becoming the default action of a parent submenu.  
 This e.g. leads to a bold label.  
 The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
 Sets the displayed text.

void **setIcon** (QString *icon*)  
 Sets the icon.

void **setEnabled** (bool *enabled*)  
 Sets if the item is enabled.

void **setChecked** (bool *checked*)  
 Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
 Sets the visibility of this item.

bool **visible** ()  
 Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
 Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
 Sets the parent action. .

void **initializeAsync** (std::function<void>  
 > *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
 Synchronously initializes the action.

bool **isInitialized** ()  
 Checks if this action is initialized.

bool **isInitializing** ()  
 Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

## Public Static Functions

void **doInitialize** ()  
void **doShutdown** ()

**class ActionTransferToCopy** : public *sh::actions::common::ActionAbstractTransferTo*  
*#include <actiontransferto.h>* Action for copying a selection to some other locations.

## Public Functions

**ActionTransferToCopy** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>>  
*nodes*)

**const** QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> **subitems** ()  
Returns the list of subitems from this submenu.

void **setSubitems** (**const** QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>>  
*subitems*)  
Sets the subitems.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [\*ActionDefaultPrecedenceValues\*](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

```

void setVisible (bool visible)
    Sets the visibility of this item.

bool visible ()
    Checks the visibility of this item (non-recursively).

std::weak_ptr<AbstractActionItem> parentAction ()
    Returns the parent action, if it is added to a container.

void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.

```

## Signals

```

void subitemsChanged ()
    Emits when the list of subitems changed.

void changed ()
    Emits when some data changed.

```

## Public Static Functions

```

void doInitialize ()
    Only called by the Shallot infrastructure for initialization.

void doShutdown ()
    Only called by the Shallot infrastructure for initialization.

```

```

class ActionTransferToHelper : public sh::actions::ActionActionItem
    #include <actiontransferto.h> This action is used as subitems in ActionAbstractTransferTo.

```

## Public Functions

```

ActionTransferToHelper (QString text, QString icon, std::shared_ptr<const
    sh::filesystem::Eurl> transferto,
    std::shared_ptr<ActionAbstractTransferTo> root)

void execute ()
    Executes this action.

void execute (sh::actions::ActionExecutionInfo *info)
    Executes this action.

QKeySequence shortcuthint ()
    Returns the keyboard shortcut for triggering this action.

```

**bool shortcutHintTriggersOnCurrentDirectory ()**  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

**void setShortcutHint (QKeySequence shortcut, bool triggersOnCurrentDirectory = false)**  
Sets the keyboard shortcut for triggering this action.

**QString text ()**  
Returns the displayed text for this action.

**QString icon ()**  
Returns the icon for this action.

**bool enabled ()**  
Checks if this action is enabled.

**bool isChecked ()**  
Checks if this action is checked (has a cross in the ui).

**bool isCheckable ()**  
Checks if this action is checkable (can have a cross in the ui).

**int defaultActionPrecedence () const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

**void setText (QString text)**  
Sets the displayed text.

**void setIcon (QString icon)**  
Sets the icon.

**void setEnabled (bool enabled)**  
Sets if the item is enabled.

**void setChecked (bool checked)**  
Sets if the item is checked (has a cross in the ui).

**void setVisible (bool visible)**  
Sets the visibility of this item.

**bool visible ()**  
Checks the visibility of this item (non-recursively).

**std::weak\_ptr<AbstractActionItem> parentAction ()**  
Returns the parent action, if it is added to a container.

**void \_setParentAction (std::shared\_ptr<AbstractActionItem> parent)**  
Sets the parent action. .

**void initializeAsync (std::function<void> > oninitialized = 0)**  
Asynchronously initializes the action.

**void initializeSync ()**  
Synchronously initializes the action.

**bool isInitialized ()**  
Checks if this action is initialized.



bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

**class ActionTransferToMove** : public *sh::actions::common::ActionAbstractTransferTo*  
*#include <actiontransferto.h>* Action for moving a selection to some other locations.

## Public Functions

**ActionTransferToMove** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>>  
*nodes*)

const QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> **subitems** ()  
Returns the list of subitems from this submenu.

void **setSubitems** (const QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>>  
*subitems*)  
Sets the subitems.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () const  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [\*ActionDefaultPrecedenceValues\*](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

`bool visible ()`  
Checks the visibility of this item (non-recursively).

`std::weak_ptr<AbstractActionItem> parentAction ()`  
Returns the parent action, if it is added to a container.

`void _setParentAction (std::shared_ptr<AbstractActionItem> parent)`  
Sets the parent action. .

`void initializeAsync (std::function<void>  
> oninitialized = 0)`  
Asynchronously initializes the action.

`void initializeSync ()`  
Synchronously initializes the action.

`bool isInitialized ()`  
Checks if this action is initialized.

`bool isInitializing ()`  
Checks if this action is initializing.

## Signals

`void subitemsChanged ()`  
Emits when the list of subitems changed.

`void changed ()`  
Emits when some data changed.

## Public Static Functions

`void doInitialize ()`

`void doShutdown ()`

### namespace mainmenu

Implementation of the main menu.

Subclasses of *sh::actions::AbstractActionItem* (and possibly some auxiliary stuff). You can find other actions in the sibling namespaces and a few very special ones at different places.

**class ActionAbout : public sh::actions::ActionActionItem**  
*#include <actionabout.h>* Action showing the Shallot about box.

## Public Functions

**ActionAbout ()**

`void execute ()`  
Executes this action.

`void execute (sh::actions::ActionExecutionInfo *info)`  
Executes this action.

`QKeySequence shortcutHint ()`  
Returns the keyboard shortcut for triggering this action.

**bool shortcutHintTriggersOnCurrentDirectory ()**  
 Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

**void setShortcutHint (QKeySequence shortcut, bool triggersOnCurrentDirectory = false)**  
 Sets the keyboard shortcut for triggering this action.

**QString text ()**  
 Returns the displayed text for this action.

**QString icon ()**  
 Returns the icon for this action.

**bool enabled ()**  
 Checks if this action is enabled.

**bool isChecked ()**  
 Checks if this action is checked (has a cross in the ui).

**bool isCheckable ()**  
 Checks if this action is checkable (can have a cross in the ui).

**int defaultActionPrecedence () const**  
 Returns the precedence for becoming the default action of a parent submenu.  
 This e.g. leads to a bold label.  
 The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

**void setText (QString text)**  
 Sets the displayed text.

**void setIcon (QString icon)**  
 Sets the icon.

**void setEnabled (bool enabled)**  
 Sets if the item is enabled.

**void setChecked (bool checked)**  
 Sets if the item is checked (has a cross in the ui).

**void setVisible (bool visible)**  
 Sets the visibility of this item.

**bool visible ()**  
 Checks the visibility of this item (non-recursively).

**std::weak\_ptr<AbstractActionItem> parentAction ()**  
 Returns the parent action, if it is added to a container.

**void \_setParentAction (std::shared\_ptr<AbstractActionItem> parent)**  
 Sets the parent action. .

**void initializeAsync (std::function<void> > oninitialized = 0)**  
 Asynchronously initializes the action.

**void initializeSync ()**  
 Synchronously initializes the action.

**bool isInitialized ()**  
 Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

**class ActionAbstractSelectNodes** : public *sh::actions::ActionActionItem*  
*#include <actionselect.h>* Abstract subclass for actions which select some nodes in the current file view.

Subclassed by *sh::actions::mainmenu::ActionInvertNodeSelection*,  
*sh::actions::mainmenu::ActionSelectAllNodes*

## Public Functions

**ActionAbstractSelectNodes** (QString *text*)

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

```

void setIcon (QString icon)
    Sets the icon.

void setEnabled (bool enabled)
    Sets if the item is enabled.

void setChecked (bool checked)
    Sets if the item is checked (has a cross in the ui).

void setVisible (bool visible)
    Sets the visibility of this item.

bool visible ()
    Checks the visibility of this item (non-recursively).

std::weak_ptr<AbstractActionItem> parentAction ()
    Returns the parent action, if it is added to a container.

void __setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.

```

## Signals

```

void changed ()
    Emits when some data changed.

```

```

class ActionApplyProfile : public sh::actions::SubmenuItem
    #include <actionapplyprofile.h> Submenu action for switching the current profile.

```

## Public Functions

```

ActionApplyProfile ()

const QList<std::shared_ptr<sh::actions::AbstractActionItem>> subitems ()
    Returns the list of subitems from this submenu.

void setSubitems (const      QList<std::shared_ptr<sh::actions::AbstractActionItem>>
                  subitems)
    Sets the subitems.

QString text ()
    Returns the displayed text for this action.

QString icon ()
    Returns the icon for this action.

bool enabled ()
    Checks if this action is enabled.

```

bool **isChecked** ()

Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>)

> *oninitialized* = 0 Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **subitemsChanged** ()

Emits when the list of subitems changed.

void **changed** ()

Emits when some data changed.

**class ActionApplyProfileX : public [sh::actions::ActionActionItem](#)**

Action for switching the current profile to a certain one.

## Public Functions

**ActionApplyProfileX** (QString *profilename*, bool *checked*)

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.  
The action with the highest value becomes the default. Only values >0 will be considered.  
See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<*AbstractActionItem*> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>

> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **changed** ()

Emits when some data changed.

**class ActionFileViewChangeIconDimension : public *sh::actions::ActionActionItem***  
*#include <actionfileviewchangeicondimension.h>* Action for changing the icon size.

Subclassed by *sh::actions::mainmenu::ActionFileViewDecreaseIconDimension*,  
*sh::actions::mainmenu::ActionFileViewIncreaseIconDimension*

## Public Functions

**ActionFileViewChangeIconDimension** (QString *name*, int *direction*, QString *icon*,  
QKeySequence *shortcut*)

void **execute** ()

Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)

Executes this action.

QKeySequence **shortcuthint** ()

Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()

Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* =  
false)

Sets the keyboard shortcut for triggering this action.

QString **text** ()

Returns the displayed text for this action.

QString **icon** ()

Returns the icon for this action.

bool **enabled** ()

Checks if this action is enabled.

bool **isChecked** ()

Checks if this action is checked (has a cross in the ui).



bool **isCheckable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>)

> *oninitialized* = 0 Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **changed** ()

Emits when some data changed.

**class ActionFileViewDecreaseIconDimension : public [sh::actions::mainmenu::ActionFileViewChange](#)**  
**#include <actionfileviewchangeicondimension.h>** Action for decreasing the icon size.

## Public Functions

**ActionFileViewDecreaseIconDimension** ()

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \*info)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

```
void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.
```

## Signals

```
void changed ()
    Emits when some data changed.
```

```
class ActionFileviewFilesizeFormatting : public sh::actions::ActionActionItem
    #include <actionfileviewfilesizeformatting.h> Action for toggling the filesize formatting mode for
    the current fileview.
```

## Public Functions

```
ActionFileviewFilesizeFormatting ()

void execute ()
    Executes this action.

void execute (sh::actions::ActionExecutionInfo *info)
    Executes this action.

QKeySequence shortcuthint ()
    Returns the keyboard shortcut for triggering this action.

bool shortcuthintTriggersOnCurrentDirectory ()
    Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the
    entry selection).

void setShortcuthint (QKeySequence shortcut, bool triggersOnCurrentDirectory =
    false)
    Sets the keyboard shortcut for triggering this action.

QString text ()
    Returns the displayed text for this action.

QString icon ()
    Returns the icon for this action.

bool enabled ()
    Checks if this action is enabled.

bool isChecked ()
    Checks if this action is checked (has a cross in the ui).

bool isCheckable ()
    Checks if this action is checkable (can have a cross in the ui).
```

int **defaultActionPrecedence** () const

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>)

> *oninitialized* = 0 Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **changed** ()

Emits when some data changed.

**class ActionFileViewIncreaseIconDimension** : public [sh::actions::mainmenu::ActionFileViewChange](#)  
*#include <actionfileviewchangeicondimension.h>* Action for increasing the icon size.

## Public Functions

**ActionFileViewIncreaseIconDimension** ()

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo \*info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<*AbstractActionItem*> parent)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

**class ActionFileviewShowHiddenFiles** : public *sh::actions::ActionActionItem*  
*#include <actionfileviewshowhiddenfiles.h>* Action which toggles the visibility of hidden files in the current fileview.

## Public Functions

**ActionFileviewShowHiddenFiles** ()

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \*info)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () const

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>)

> *oninitialized* = 0 Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **changed** ()

Emits when some data changed.

**class ActionFileviewView: public [sh::actions::ActionActionItem](#)**

*#include <actionfileviewview.h>* Action which toggles between icon- and listmode for the current fileview.

## Public Functions

**ActionFileviewView()**

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo \*info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.



```
void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.
```

## Signals

```
void changed ()
    Emits when some data changed.
```

```
class ActionGotoDirectory : public sh::actions::ActionActionItem
    #include <actiongotodirectory.h> Action which toggles the visibility of hidden files in the current
    fileview.
```

## Public Functions

```
ActionGotoDirectory ()

void execute ()
    Executes this action.

void execute (sh::actions::ActionExecutionInfo *info)
    Executes this action.

QKeySequence shortcuthint ()
    Returns the keyboard shortcut for triggering this action.

bool shortcuthintTriggersOnCurrentDirectory ()
    Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the
    entry selection).

void setShortcuthint (QKeySequence shortcut, bool triggersOnCurrentDirectory =
    false)
    Sets the keyboard shortcut for triggering this action.

QString text ()
    Returns the displayed text for this action.

QString icon ()
    Returns the icon for this action.

bool enabled ()
    Checks if this action is enabled.

bool isChecked ()
    Checks if this action is checked (has a cross in the ui).

bool isCheckable ()
    Checks if this action is checkable (can have a cross in the ui).
```

int **defaultActionPrecedence** () const

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>)

> *oninitialized* = 0 Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **changed** ()

Emits when some data changed.

**class ActionHelp**: public [sh::actions::ActionActionItem](#)

*#include <actionhelp.h>* Action for opening the Shallot help system.

## Public Functions

**ActionHelp** ()

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo \*info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

```
void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.
```

## Signals

```
void changed ()
    Emits when some data changed.
```

```
class ActionInvertNodeSelection : public sh::actions::mainmenu::ActionAbstractSelectNodes
    #include <actionselect.h> Action for inverting the node selection in the current file view.
```

## Public Functions

```
ActionInvertNodeSelection ()

void execute ()
    Executes this action.

void execute (sh::actions::ActionExecutionInfo *info)
    Executes this action.

QKeySequence shortcuthint ()
    Returns the keyboard shortcut for triggering this action.

bool shortcuthintTriggersOnCurrentDirectory ()
    Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the
    entry selection).

void setShortcuthint (QKeySequence shortcut, bool triggersOnCurrentDirectory =
    false)
    Sets the keyboard shortcut for triggering this action.

QString text ()
    Returns the displayed text for this action.

QString icon ()
    Returns the icon for this action.

bool enabled ()
    Checks if this action is enabled.

bool isChecked ()
    Checks if this action is checked (has a cross in the ui).

bool isCheckable ()
    Checks if this action is checkable (can have a cross in the ui).
```

int **defaultActionPrecedence** () const

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>)

> *oninitialized* = 0 Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **changed** ()

Emits when some data changed.

## Private Members

QList<std::shared\_ptr<[sh::filesystem::FilesystemNode](#)>> **wasselected**

**class ActionMain** : public [sh::actions::SubmenuActionItem](#), public [sh::base::Singleton](#)

#include <actionmain.h> Submenu action providing the Shallot main menu.

## Public Functions

**const** QList<std::shared\_ptr<[sh::actions::AbstractActionItem](#)>> **subitems** ()

Returns the list of subitems from this submenu.

void **setSubitems** (**const**       QList<std::shared\_ptr<[sh::actions::AbstractActionItem](#)>>  
                                  *subitems*)

Sets the subitems.

QString **text** ()

Returns the displayed text for this action.

QString **icon** ()

Returns the icon for this action.

bool **enabled** ()

Checks if this action is enabled.

bool **isChecked** ()

Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>

> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

**bool isInitializing ()**  
Checks if this action is initializing.

**void doInitialize ()**  
Executes singleton initialization.

**void doShutdown ()**  
Executes singleton shutdown.

**void shutdown ()**  
Shutdown down this singleton.

**bool isAlive ()**  
Returns if this singleton is alive (true until its shutdown begins).

## Signals

**void subitemsChanged ()**  
Emits when the list of subitems changed.

**void changed ()**  
Emits when some data changed.

## Private Functions

**ActionMain ()**

**class ActionManageSavedSettings : public *sh::actions::ActionActionItem***  
*#include <actionmanagesavedsettings.h>* Action for opening the 'Manage saved settings' dialog.

## Public Functions

**ActionManageSavedSettings ()**

**void execute ()**  
Executes this action.

**void execute (*sh::actions::ActionExecutionInfo* \*info)**  
Executes this action.

**QKeySequence shortcutHint ()**  
Returns the keyboard shortcut for triggering this action.

**bool shortcutHintTriggersOnCurrentDirectory ()**  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

**void setShortcutHint (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)**  
Sets the keyboard shortcut for triggering this action.

**QString text ()**  
Returns the displayed text for this action.

**QString icon ()**  
Returns the icon for this action.

**bool enabled ()**  
Checks if this action is enabled.

bool **isChecked** ()

Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>)

> *oninitialized* = 0 Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **changed** ()

Emits when some data changed.

**class ActionNumberFileviews : public [sh::actions::SubmenuActionItem](#)**

*#include <actionnumberfileviews.h>* Submenu action for changing the number of fileviews.



## ActionNumberFileviews ()

```
const OList<std::shared_ptr<sh::actions::AbstractActionItem>> subitems()
```

```
void setSubitems (const QList<std::shared_ptr<sh::actions::AbstractActionItem>>  
                  subitems)
```

QString **text** ()

QString **icon** ()

```
bool enabled()
```

```
bool isChecked()
```

```
bool isCheckable ()
```

```
int defaultActionPrecedence() const
```

This e.g. leads to a bold label.

```
void setText (QString text)
```

```
void setIcon (QString icon)
```

```
void setEnabled (bool enabled)
```

```
void setChecked (bool checked)
```

```
void setVisible (bool visible)
```

```
bool visible ()
```

```
std::weak_ptr<AbstractActionItem> parentAction ()
```

```
void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
```

```
void initializeAsync (std::function<void>)
```

```
void initializeSync ()
```

701

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **subitemsChanged** ()  
Emits when the list of subitems changed.

void **changed** ()  
Emits when some data changed.

## Friends

**friend class** ActionNumberFileviews\_Add

**friend class** ActionNumberFileviews\_Remove

**friend class** ActionNumberFileviews\_Remove\_Current

**class ActionNumberFileviews\_Add**: public *sh::actions::ActionActionItem*  
*#include <actionnumberfileviews.h>* Action for adding a new fileview.

## Public Functions

**ActionNumberFileviews\_Add** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>>  
nodes = {})

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \*info)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence shortcut, bool triggersOnCurrentDirectory =  
false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckedable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>

> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **changed** ()

Emits when some data changed.

## Public Static Functions

void **doInitialize** ()

void **doShutdown** ()

**class ActionNumberFileviews\_Remove : public *sh::actions::ActionActionItem***  
*#include <actionnumberfileviews.h>* Action for removing a certain fileview.

## Public Functions

**ActionNumberFileviews\_Remove** (int *i*, bool *deflt*)

void **execute** ()

Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)

Executes this action.

QKeySequence **shortcuthint** ()

Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()

Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)

Sets the keyboard shortcut for triggering this action.

QString **text** ()

Returns the displayed text for this action.

QString **icon** ()

Returns the icon for this action.

bool **enabled** ()

Checks if this action is enabled.

bool **isChecked** ()

Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [\*ActionDefaultPrecedenceValues\*](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

```

void setChecked (bool checked)
    Sets if the item is checked (has a cross in the ui).

void setVisible (bool visible)
    Sets the visibility of this item.

bool visible ()
    Checks the visibility of this item (non-recursively).

std::weak_ptr<AbstractActionItem> parentAction ()
    Returns the parent action, if it is added to a container.

void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.

```

## Signals

```

void changed ()
    Emits when some data changed.

```

```

class ActionNumberFileviews_Remove_Current : public sh::actions::ActionActionItem
    #include <actionnumberfileviews.h> Action for removing the current fileview (for keyboard
    shortcut).

```

## Public Functions

```

ActionNumberFileviews_Remove_Current (QList<std::shared_ptr<sh::filesystem::FileSystemNode>>
    nodes)

void execute ()
    Executes this action.

void execute (sh::actions::ActionExecutionInfo *info)
    Executes this action.

QKeySequence shortcuthint ()
    Returns the keyboard shortcut for triggering this action.

bool shortcuthintTriggersOnCurrentDirectory ()
    Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the
    entry selection).

void setShortcuthint (QKeySequence shortcut, bool triggersOnCurrentDirectory =
    false)
    Sets the keyboard shortcut for triggering this action.

QString text ()
    Returns the displayed text for this action.

```

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

## Public Static Functions

void **doInitialize** ()

void **doShutdown** ()

**class ActionQuit** : public *sh::actions::ActionActionItem*  
*#include <actionquit.h>* Action for closing Shallot.

## Public Functions

**ActionQuit** ()

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \*info)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence shortcut, bool triggersOnCurrentDirectory = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString text)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<AbstractActionItem> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_\_setParentAction** (std::shared\_ptr<AbstractActionItem> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0)  
Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

**class ActionRefresh** : public *sh::actions::ActionActionItem*  
*#include <actionrefresh.h>* Action for refreshing the current view.

## Public Functions

**ActionRefresh** ()

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.



**QString text ()**  
Returns the displayed text for this action.

**QString icon ()**  
Returns the icon for this action.

**bool enabled ()**  
Checks if this action is enabled.

**bool isChecked ()**  
Checks if this action is checked (has a cross in the ui).

**bool isCheckable ()**  
Checks if this action is checkable (can have a cross in the ui).

**int defaultActionPrecedence () const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

**void setText (QString text)**  
Sets the displayed text.

**void setIcon (QString icon)**  
Sets the icon.

**void setEnabled (bool enabled)**  
Sets if the item is enabled.

**void setChecked (bool checked)**  
Sets if the item is checked (has a cross in the ui).

**void setVisible (bool visible)**  
Sets the visibility of this item.

**bool visible ()**  
Checks the visibility of this item (non-recursively).

**std::weak\_ptr<AbstractActionItem> parentAction ()**  
Returns the parent action, if it is added to a container.

**void \_setParentAction (std::shared\_ptr<AbstractActionItem> parent)**  
Sets the parent action. .

**void initializeAsync (std::function<void> > oninitialized = 0)**  
Asynchronously initializes the action.

**void initializeSync ()**  
Synchronously initializes the action.

**bool isInitialized ()**  
Checks if this action is initialized.

**bool isInitializing ()**  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

**class ActionSaveSettings : public [sh::actions::ActionActionItem](#)**  
*#include <actionsavesettings.h>* Action for opening the ‘Save settings’ dialog.

## Public Functions

**ActionSaveSettings** ()

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo \*info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

```

void setChecked (bool checked)
    Sets if the item is checked (has a cross in the ui).

void setVisible (bool visible)
    Sets the visibility of this item.

bool visible ()
    Checks the visibility of this item (non-recursively).

std::weak_ptr<AbstractActionItem> parentAction ()
    Returns the parent action, if it is added to a container.

void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.

```

## Signals

```

void changed ()
    Emits when some data changed.

```

```

class ActionSearch : public sh::actions::ActionActionItem
    #include <actionsearch.h> Action for opening the 'Save settings' dialog.

```

## Public Functions

```

ActionSearch ()

void execute ()
    Executes this action.

void execute (sh::actions::ActionExecutionInfo *info)
    Executes this action.

QKeySequence shortcuthint ()
    Returns the keyboard shortcut for triggering this action.

bool shortcuthintTriggersOnCurrentDirectory ()
    Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the
    entry selection).

void setShortcuthint (QKeySequence shortcut, bool triggersOnCurrentDirectory =
    false)
    Sets the keyboard shortcut for triggering this action.

QString text ()
    Returns the displayed text for this action.

QString icon ()
    Returns the icon for this action.

```

bool **enabled** ()

Checks if this action is enabled.

bool **isChecked** ()

Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>

> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

**class ActionSelectAllNodes** : public *sh::actions::mainmenu::ActionAbstractSelectNodes*  
*#include <actionselect.h>* Action for selecting all nodes in the current file view.

## Public Functions

**ActionSelectAllNodes** ()

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \*info)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence shortcut, bool triggersOnCurrentDirectory = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString text)  
Sets the displayed text.

void **setIcon** (QString icon)  
Sets the icon.

void **setEnabled** (bool enabled)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<*AbstractActionItem*> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<*AbstractActionItem*> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0)  
Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

**class ActionSetWindowTitlePattern** : public *sh::actions::ActionActionItem*  
*#include <actionsetwindowtitlepattern.h>* Action for setting the window title pattern.

## Public Functions

**ActionSetWindowTitlePattern** ()

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
 Checks if this action is enabled.

bool **isChecked** ()  
 Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
 Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
 Returns the precedence for becoming the default action of a parent submenu.  
 This e.g. leads to a bold label.  
 The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
 Sets the displayed text.

void **setIcon** (QString *icon*)  
 Sets the icon.

void **setEnabled** (bool *enabled*)  
 Sets if the item is enabled.

void **setChecked** (bool *checked*)  
 Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
 Sets the visibility of this item.

bool **visible** ()  
 Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
 Returns the parent action, if it is added to a container.

void **\_\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
 Sets the parent action. .

void **initializeAsync** (std::function<void>  
 > *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
 Synchronously initializes the action.

bool **isInitialized** ()  
 Checks if this action is initialized.

bool **isInitializing** ()  
 Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

**class ActionShowDetailPanel** : public [\*sh::actions::ActionActionItem\*](#)  
*#include <actionshowdetailpanel.h>* Action for toggling the visibility of the detail panel.

## Public Functions

**ActionShowDetailPanel** ()

void **execute** ()  
Executes this action.

void **execute** ([\*sh::actions::ActionExecutionInfo\*](#) \*info)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence shortcut, bool triggersOnCurrentDirectory = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [\*ActionDefaultPrecedenceValues\*](#) for reference values.

void **setText** (QString text)  
Sets the displayed text.

void **setIcon** (QString icon)  
Sets the icon.

void **setEnabled** (bool enabled)  
Sets if the item is enabled.



```

void setChecked (bool checked)
    Sets if the item is checked (has a cross in the ui).

void setVisible (bool visible)
    Sets the visibility of this item.

bool visible ()
    Checks the visibility of this item (non-recursively).

std::weak_ptr<AbstractActionItem> parentAction ()
    Returns the parent action, if it is added to a container.

void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.

```

## Signals

```

void changed ()
    Emits when some data changed.

```

```

class ActionShowFolderTree : public sh::actions::ActionActionItem
    #include <actionshowfoldertree.h> Action for toggling the visibility of the directory tree.

```

## Public Functions

```

ActionShowFolderTree ()

void execute ()
    Executes this action.

void execute (sh::actions::ActionExecutionInfo *info)
    Executes this action.

QKeySequence shortcuthint ()
    Returns the keyboard shortcut for triggering this action.

bool shortcuthintTriggersOnCurrentDirectory ()
    Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the
    entry selection).

void setShortcuthint (QKeySequence shortcut, bool triggersOnCurrentDirectory =
    false)
    Sets the keyboard shortcut for triggering this action.

QString text ()
    Returns the displayed text for this action.

QString icon ()
    Returns the icon for this action.

```

`bool enabled ()`  
Checks if this action is enabled.

`bool isChecked ()`  
Checks if this action is checked (has a cross in the ui).

`bool isCheckable ()`  
Checks if this action is checkable (can have a cross in the ui).

`int defaultActionPrecedence () const`  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

`void setText (QString text)`  
Sets the displayed text.

`void setIcon (QString icon)`  
Sets the icon.

`void setEnabled (bool enabled)`  
Sets if the item is enabled.

`void setChecked (bool checked)`  
Sets if the item is checked (has a cross in the ui).

`void setVisible (bool visible)`  
Sets the visibility of this item.

`bool visible ()`  
Checks the visibility of this item (non-recursively).

`std::weak_ptr<AbstractActionItem> parentAction ()`  
Returns the parent action, if it is added to a container.

`void _setParentAction (std::shared_ptr<AbstractActionItem> parent)`  
Sets the parent action. .

`void initializeAsync (std::function<void>  
> oninitialized = 0)`  
Asynchronously initializes the action.

`void initializeSync ()`  
Synchronously initializes the action.

`bool isInitialized ()`  
Checks if this action is initialized.

`bool isInitializing ()`  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

**class ActionShowLog** : public *sh::actions::ActionActionItem*  
*#include <actionshowlog.h>* Action for showing the Shallot log in a dialog.

## Public Functions

**ActionShowLog** ()

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \*info)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<*AbstractActionItem*> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_\_setParentAction** (std::shared\_ptr<*AbstractActionItem*> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

**class ActionTreeStickyness** : public *sh::actions::ActionActionItem*  
*#include <actiontreestickyness.h>* Action for toggling the stickyness of the directory tree to the fileviews.

## Public Functions

**ActionTreeStickyness** ()

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

**class ActionTuning**: public *sh::actions::ActionActionItem*  
*#include <actiontuning.h>* Action for opening the ‘Tuning’ dialog.

## Public Functions

**ActionTuning** ()

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \*info)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence shortcut, bool triggersOnCurrentDirectory = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString text)  
Sets the displayed text.

void **setIcon** (QString icon)  
Sets the icon.

void **setEnabled** (bool enabled)  
Sets if the item is enabled.

```

void setChecked (bool checked)
    Sets if the item is checked (has a cross in the ui).

void setVisible (bool visible)
    Sets the visibility of this item.

bool visible ()
    Checks the visibility of this item (non-recursively).

std::weak_ptr<AbstractActionItem> parentAction ()
    Returns the parent action, if it is added to a container.

void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.

```

## Signals

```

void changed ()
    Emits when some data changed.

```

## namespace base

Core logic.

Includes common data structures and some basic infrastructure. Some other namespaces in sh contain parts of the infrastructure as well.

## Enums

### enum LogSeverity

The severity of a log message.

*Values:*

```

enumerator DEBUG = 0
enumerator INFO = 100
enumerator _DEFAULT = INFO
enumerator WARNING = 200
enumerator ERROR = 300
enumerator E_ERROR = ERROR

```

```

class IconManager : public QObject, public sh::base::Singleton
    #include <iconmanager.h> Fetches icons according to theme availability and settings.

```

## Public Functions

QIcon **getIcon** (QString *mainname*, QString *emblemname* = QString(), QString *miniemblemname* = QString(), QStringList *tags* = QStringList())  
Creates a QIcon from the icon name(s). Uses a cache.

QIcon **getIconByFullname** (QString *fullname*)  
Creates a QIcon from the icon fullname. Uses a cache.

QIcon **getIcon** (QIcon *mainicon*, QString *emblemname*, QString *miniemblemname*, QStringList *tags*)  
Creates a QIcon based on an existing one. Uncached.

QPixmap **getPixmap** (QString *name*, int *size* = 22)  
Creates a QPixmap for an icon name. Uncached.

**~IconManager** ()

void **doInitialize** ()  
Executes singleton initialization.

void **doShutdown** ()  
Executes singleton shutdown.

void **shutdown** ()  
Shutdown down this singleton.

bool **isAlive** ()  
Returns if this singleton is alive (true until its shutdown begins).

## Private Functions

**IconManager** ()

QIcon **\_getIcon** (QString *name*)

QImage **\_colourImage** (QImage *img*, QColor *color*)

## Private Members

QMutex **cachemutex**

QHash<QString, QIcon> **cache**

QList<GetIconStrategy\*> **getIconStrategies**

std::shared\_ptr<sh::configuration::ConfigurationValue> **cfgvalPreferredStrategy**

**class GetIconStrategy**

Subclassed by *sh::base::IconManager::IncludedGetIconStrategy*,  
*sh::base::IconManager::QIconFromThemeGetIconStrategy*



### Public Functions

QIcon **getIcon** (QString *name*) = 0

**~GetIconStrategy** ()

**class** **IncludedGetIconStrategy** : **public** *sh::base::IconManager::GetIconStrategy*

### Public Functions

**IncludedGetIconStrategy** ()

QIcon **getIcon** (QString *name*)

### Private Members

QColor **brandingcolor1**

QColor **brandingcolor2**

**class** **QIconFromThemeGetIconStrategy** : **public** *sh::base::IconManager::GetIconStrategy*

### Public Functions

**QIconFromThemeGetIconStrategy** ()

QIcon **getIcon** (QString *name*)

### Private Members

QHash<QString, QString> **\_aliases**

**class** **Logger** : **public** QObject, **public** *sh::base::Singleton*

*#include <logger.h>* The logging manager.

Use it for writing messages to the Shallot log and for reading from it.

Note: Logging is easiest by the LOG\_\* macros like SH\_LOG\_INFO (and it provides more meta data!).

### Public Functions

void **logException** (*sh::exceptions::Exception* &*ex*, *LogSeverity* *severity* = *LogSeverity::ERROR*, QString *source* = QString())

Logs an exception.

void **log** (QString *message*, *LogSeverity* *severity*, QString *source* = QString())

Logs a message.

QString **getLogAsText** (*LogSeverity* *minseverity* = *LogSeverity::DEBUG*)

Returns all logged messages as one text.

QList<*LogMessage\**> **logMessages** ()

Returns the list of logged messages.

QString **logPrefix** ()

Returns the log prefix, i.e. an optional string all log output begins with.

**~Logger ()**  
void **doInitialize ()**  
    Executes singleton initialization.  
void **doShutdown ()**  
    Executes singleton shutdown.  
void **shutdown ()**  
    Shutdown down this singleton.  
bool **isAlive ()**  
    Returns if this singleton is alive (true until its shutdown begins).

## Signals

void **newLogMessageArrived ()**  
    Emitted when a new log message was written.

## Private Functions

**Logger ()**

## Private Members

QMutex **mutex**  
QList<*LogMessage\**> **\_logMessages**  
QString **\_logPrefix**  
**struct LogMessage**  
    *#include <logger.h>* A single log message.

## Public Members

QString **message**  
*LogSeverity* **severity**  
QString **source**  
QDateTime **time**  
**class MainThread**  
    *#include <mainthread.h>* The main thread.  
    Lots of data structure may only accessed from the main thread. The UI and the *sh::filesystem::FilesystemModel* also live in this thread (although some functions there may allow multi-threading in some ways).

## Public Static Functions

*sh::base::ThreadDispatcher* \*dispatcher ()

The *sh::base::ThreadDispatcher*, which allows to execute code in the main thread.

**class ShallotProcess : public *sh::base::Singleton***

#include <shallotprocess.h> Provides parameters given from the user by command line and some more simple infos.

## Public Functions

QString **uiMode** ()

The ui mode (e.g. “qt”, “web”) specified on command line.

QMap<QString, QString> **configurationValueAssignments** ()

The configuration value assignments set on command line.

QList<QString> **configurationValueFiles** ()

The configuration value files set on command line.

QString **parameterValue** (QString *key*, QString *deflt* = QString())

Returns the command line parameter value for a key as string.

QStringList **parameterValueList** (QString *key*)

Returns the command line parameter value for a key as list of strings (for multiple parameter usage).

qint64 **parameterValueInt** (QString *key*, qint64 *deflt* = 0)

Returns the command line parameter value for a key as integer.

QStringList **parameters** ()

Returns the list of keys of all specified command line parameter names.

QString **initialWorkDirectory** ()

Returns the initial directory specified on command line.

QString **logPrefix** ()

Returns the log prefix (i.e. an optional string all log output begins with) specified on command line.

QStringList **lang** ()

Returns the ui language specified on command line.

QColor **brandingColor** ()

Returns the Shallot branding color (typically a dark red).

int **minport** ()

Return the minimal allowed port specified on command line (for web ui).

int **maxport** ()

Return the maximal allowed port specified on command line (for web ui).

int **workerminport** ()

Return the minimal allowed port specified on command line for spawning web workers with (for web ui).

int **workermaxport** ()

Return the maximal allowed port specified on command line for spawning web workers with (for web ui).

int **maxnumberworkers** (int *deflt*)  
Maximum number of workers specified on command line.

int **maxnumberworkersperhost** (int *deflt*)  
Maximum number of workers per host specified on command line.

int **maxmemorypercent** (int *deflt*)  
Maximum memory in percent specified on command line.

int **maxmemorypercentglobal** (int *deflt*)  
Maximum memory in percent globally specified on command line.

int **maxbrowserawayseconds** (int *deflt*)  
Maximum browser away time in seconds specified on command line.

int **maxidlenessseconds** (int *deflt*)  
Maximum user idleness time in seconds specified on command line.

void **doInitialize** ()  
Executes singleton initialization.

void **doShutdown** ()  
Executes singleton shutdown.

void **shutdown** ()  
Shutdown down this singleton.

bool **isAlive** ()  
Returns if this singleton is alive (true until its shutdown begins).

## Public Static Functions

QString **userDataDir** ()  
Returns the path to a storage for per-user data. It resides somewhere within the user home directory.

QString **shallotDataDir** ()  
Returns the directory of the Shallot program data. It contains the static files which are part of the Shallot program.

QString **revisionString** ()  
Returns the Shallot revision string, i.e. the exact version number. Note: This is only updated by a special build tool (anise) and not by directly compiling via qmake.

QString **homepage** ()  
Returns the url to the Shallot homepage.

QDateTime **buildtime** ()  
Returns the time when this Shallot build was created. Note: This is only updated by a special build tool (anise) and not by directly compiling via qmake.

## Private Functions

**ShallotProcess ()**

## Private Members

QMutex **\_brandingcolormutex**

QMap<QString, QStringList> **\_cmdlineargs**

QString **\_workdir**

QString **\_ui**

QMap<QString, QString> **\_cfgvalassignments**

QList<QString> **\_cfgvalfiles**

QColor **\_brandingcolor**

std::shared\_ptr<sh::configuration::ConfigurationValue> **cfgvalBrandingColor**

int **\_minport** = 0

int **\_maxport** = 0

int **\_workerminport** = 0

int **\_workermaxport** = 0

## class Singleton

*#include <singletoninitializer.h>* A singleton with initialization on Shallot startup and shutdown in the end.

See *SingletonInitializer*.

Subclassed by *sh::actions::ActionsManager*, *sh::actions::mainmenu::ActionMain*,  
*sh::base::IconManager*, *sh::base::Logger*, *sh::base::ShallotProcess*,  
*sh::configuration::ConfigurationManager*, *sh::detailcolumns::DetailColumnCustomAttributes*,  
*sh::detailcolumns::DetailColumnDirectSymlinkTarget*, *sh::detailcolumns::DetailColumnExtendedAttributes*,  
*sh::detailcolumns::DetailColumnFilesize*, *sh::detailcolumns::DetailColumnMimetype*,  
*sh::detailcolumns::DetailColumnMtime*, *sh::detailcolumns::DetailColumnResolvedSymlink*,  
*sh::exceptions::ExceptionHandlerSettingsManager*, *sh::filesystem::FilesystemHandlerRegister*,  
*sh::filesystem::FilesystemModel*, *sh::filesystem::FilesystemModelDirectoryTreeProxy*,  
*sh::filesystem::FilesystemModelDirectoryTreeProxyVisibilityEnforcements*,  
*sh::filesystemhandlers::GnomeIODevicesFilesystemHandler*, *sh::filesystemhandlers::GnomeIONetworkFilesystemHan*,  
*sh::filesystemhandlers::GnomeIOSmbFilesystemHandler*, *sh::filesystemhandlers::LocalFilesystemHandler*,  
*sh::filesystemhandlers::SharcFilesystemHandler*, *sh::paneldetails::PanelDetailManager*,  
*sh::scripting::api::ApiGlobalObject*, *sh::scripting::PythonScriptInterpreter*,  
*sh::scripting::ScriptingEngine*, *sh::search::SearchFilesystemHandler*, *sh::search::SearchManager*,  
*sh::settings::SettingsManager*, *sh::tools::accounts::AccountsManager*, *sh::tools::Benchmarking*,  
*sh::tools::BookmarkManager*, *sh::tools::DataExchange*, *sh::tools::filetypes::FileTypeManager*,  
*sh::tools::filetypes::UserDefinedOpenMethodDeterminationStrategy*,  
*sh::tools::LocalFilesystemWatcher*, *sh::tools::LocalFilesystemWatcherConnector*,  
*sh::tools::OperationsCache*, *sh::tools::ThumbnailManager*, *sh::tools::VisibleViews*

## Public Functions

```
void doInitialize ()  
    Executes singleton initialization.  
  
void doShutdown ()  
    Executes singleton shutdown.  
  
void shutdown ()  
    Shutdown down this singleton.  
  
bool isAlive ()  
    Returns if this singleton is alive (true until its shutdown begins).  
  
~Singleton ()  
  
Singleton (const Singleton&) = delete  
Singleton (Singleton&&) = delete
```

## Private Members

```
QMutex _mutex_shutdown  
  
bool _shutdown = false
```

### **class SingletonInitializer**

*#include <singletoninitializer.h>* Takes care of initialization and shutdown of infrastructure singletons.

It is a very early part of shallot core infrastructure. Singletons typically participate by using those macros:

**DECLARE\_SINGLETON**(STYPE): Used inside the singleton class definition. STYPE is the singleton's class. It must be a subclass of *sh::base::Singleton*. It will get a static `std::shared_ptr<STYPE> instance()` method by that.

**REGISTER\_SINGLETON**(NS, STYPE, GROUPNAME): Used inside the source file. NS is the namespace of the singleton. STYPE is the singleton's class. GROUPNAME is the singleton group name (used for dependency handling, see later).

For singleton groups, there is **REGISTER\_SINGLETON\_GROUP**(GROUPNAME, ...): GROUPNAME is the group name used for grouping singletons together. Additional arguments are other group names; those groups are considered as dependencies, which must be fulfilled before this group can.

For just initializing stuff via static methods (without a singleton instance), use **REGISTER\_STATICINIT**(NS, STYPE, GROUPNAME): NS is the namespace of the class to initialize. STYPE is the class to initialize. It must provide public static void `doInitialize()` and static void `doShutdown()`. GROUPNAME is the group name.

## Public Functions

void **callInitializers** ()

Executes all create-callbacks at first, then all init-callbacks. In both runs, the callbacks with lower index come first. Afterwards, the singletons are considered as up and running.

**SingletonInitializer** ()

Constructed only by the infrastructure and made available otherwise.

**~SingletonInitializer** ()

void **shutdown** ()

Executes all shutdown-callbacks at first, then all remove-callbacks. The callback order is the reversed one of callInitializers. Afterwards, the singletons are considered as shut down and removed.

## Public Static Functions

char **registerGroup** (QString *groupname*, QStringList *groupdeps*)

Registers a singleton group.

You should typically use REGISTER\_SINGLETON\_GROUP. See [SingletonInitializer](#).

char **registerSingleton** (QString *name*, std::function<std::shared\_ptr<Singleton>()> *instancefct*, QString *groupname*)

Registers a singleton.

You should typically use DECLARE\_SINGLETON and REGISTER\_SINGLETON. See [SingletonInitializer](#).

QThread \***initializerThread** ()

bool **isShutdown** ()

If the singletons are completely shut down and removed.

## Private Types

```
std::tuple< QString, QString, std::function< std::shared_ptr< Singleton >> > > S
typedef std::tuple<QString, QStringList> SingletonGroupTuple
```

## Private Static Functions

bool **dependsOn** (QString *group*, QString *depgroup*)

## Private Static Attributes

QList<SingletonTuple> \***\_singletons**

QHash<QString, [SingletonGroupTuple](#)> \***\_groups**

bool **\_isshutdown** = false

QMutex **shutdownmutex**

**class ThreadDispatcher** : public QObject

*#include <threaddispatcher.h>* Dispatcher for sync/async invocation of functions into the associated thread.

## Public Functions

**ThreadDispatcher** (QObject *\*parent*, QThread *\*thread*)

Constructed only by the infrastructure and made available otherwise.

void **invokeSync** (std::function<void>)

>Synchronously executes a function in the dispatcher's thread.

If it already runs in the current thread, it just executes the function. It returns when the function is completely executed.

void **invokeAsync** (std::function<void>)

>Asynchronously executes a function in the dispatcher's thread.

It directly returns. The actual execution will take place in a later message loop iteration of that thread.

void **invokeAsync** (std::function<void>)

>std::shared\_ptr<void> *obj1*, std::shared\_ptr<void> *obj2* = 0, std::shared\_ptr<void> *obj3* = 0Like the other variant, but it can also conserve some shared pointers until after execution.

bool **inThread** ()

Returns if the current thread is already equal to the dispatcher's thread (so dispatching isn't required).

## Signals

void **\_invoked** ()

## Private Functions

void **\_invokeAsync** (std::function<void>)

>

## Private Members

QMutex **callsmutex**

QThread **\*\_thread**

QQueue<std::function<void ()>> **\_queue**

## Private Slots

void **slot\_invoked** ()

**class ThreadPool** : public QObject

*#include <threadpool.h>* A pool of worker threads doing some short jobs from a queue.



## Public Static Functions

void **enqueueForce** (std::function<void>  
> *fcn*) Enqueues code to the threadpool.

void **enqueueForce** (std::function<void>  
> *fcn*, std::shared\_ptr<void> *obj1*, std::shared\_ptr<void> *obj2* = 0, std::shared\_ptr<void> *obj3* = 0) Enqueues code to the threadpool.

It can also conserve some shared pointers until after execution.

void **enqueueForceBeyondAsyncCallBarrier** (std::function<void>  
> *fcn*, std::shared\_ptr<void> *obj1* = 0, std::shared\_ptr<void> *obj2* = 0, std::shared\_ptr<void> *obj3* = 0) Enqueues code to the threadpool and forbids async calls within it. This is useful for ensuring that no code might access a certain object after its deletion.

It can also conserve some shared pointers until after execution.

void **enqueueIfIn** (std::function<void>  
> *fcn*, QThread \**thread*) Enqueues code to the threadpool if currently in *thread*. Otherwise executes directly.

void **enqueueIfInMain** (std::function<void>  
> *fcn*) Enqueues code to the threadpool if currently in main thread. Otherwise executes directly.

void **doShutdown** ()  
Only called by the Shallot infrastructure for shutdown.

bool **isShuttingDown** ()

bool **isShutdown** ()

int **getThreadCount** ()

void **respectThreadAbort** ()

void **doInitialize** ()

## Private Static Functions

void **\_enqueueForce** (std::function<void>  
> *fcn*)

## Private Static Attributes

const int **THREAD\_COUNT** = 10

QMutex **poolmutex**

QWaitCondition **pooladdedcondition**

QList<std::function<void ()>> **tasks**

bool **\_shutdown** = false

QList<ThreadPoolThread\*> **\_threads**

int **\_threadsalive** = 0

## Friends

```
friend class ThreadPoolThread

class ThreadPoolThread: public QThread
    #include <threadpool.h> A thread in the thread pool.
```

## Friends

```
friend class ThreadPool

namespace configuration
    Configuration, so everything you see in ‘Finetuning’, and some other data (e.g. get shallot program directory).
```

## Enums

```
enum ConfigurationCategory
    Categories of Configursh::configuration::ConfigurationValuentations. They are used for grouping them in the dialogs. There is no difference in behavior implied by this choice.
```

*Values:*

```
enumerator CategoryNone = 0
    No category.

enumerator CategoryGUI = 1
    User interface category.

enumerator CategoryBehavior = 2
    Behavioral configuration category.

enumerator CategoryExternalTools = 3
    External tools category.
```

```
class ConfigurationManager: public QObject, public sh::base::Singleton
    #include <configurationmanager.h> Manages the Shallot configuration.
```

This is the more static part of Shallot. Much more stuff, i.e. everything you see in ‘Manage saved settings’, is in *sh::settings::SettingsManager*.

## Public Functions

```
std::shared_ptr<ConfigurationValue> registerConfigValue (QString name, QVariant default, ConfigurationValueType *valuetype = 0, QString desc = "", ConfigurationCategory cat = CategoryNone, QString longdesc = "", QString changehint = "")
```

Creates and registers a new *ConfigurationValue*. This is typically done once at the beginning. Each registered instance is shown in the ‘Tuning’ dialog and can be set (for making changes) and observed (for applying changes) in code as well. Useful e.g. for some internal bookkeeping or for exotic machine-wide configuration values (path to some tool, ...).

```
QList<std::shared_ptr<ConfigurationValue>> getAllConfigurationValues ()
    Returns a list of all registered configuration value instances.
```

```
void doInitialize ()
    Executes singleton initialization.

void doShutdown ()
    Executes singleton shutdown.

void shutdown ()
    Shutdown down this singleton.

bool isAlive ()
    Returns if this singleton is alive (true until its shutdown begins).
```

### Private Functions

```
QVariant getFixedConfigValue (QString key)
    Returns a non-null value if 'key' was set in a fixed way (e.g. via command line).

ConfigurationManager ()
```

### Private Members

```
QHash<QString, std::shared_ptr<ConfigurationValue>> _configvalues
QHash<QString, QVariant> _fixed_configvalues
bool _fixed_configvalues_initialized = false
QMutex _mutex

class ConfigurationValueImpl : public sh::configuration::ConfigurationValue
```

### Public Functions

```
ConfigurationValueImpl (QString name, QVariant deflt, QString desc, QString
                        longdesc, ConfigurationCategory cat, ConfigurationValue-
                        Type *valuetype, QString changehint, ConfigurationMan-
                        ager *mgr)

void setConfigValue (QVariant value) override
    Sets a new configuration value.

QVariant getConfigValueVariant () override
    Returns the current value of this instance as variant.

bool mayChange () override
    Returns if changes are allowed.

int getConfigValueInt ()
    Returns the current value of this instance as integer.

double getConfigValueFloat ()
    Returns the current value of this instance as floating point number.

bool getConfigValueBool ()
    Returns the current value of this instance as boolean.

QString getConfigValueString ()
    Returns the current value of this instance as string.
```

void **consumeValue** (QObject \*o, std::function<void>  
> h) Adds a consumer function, triggered directly and whenever the value changes.

#### Parameters

- o: For lifetime monitoring.

QString **name** ()

Returns the internal name of this configuration.

QString **description** ()

Returns the short description of this configuration.

QString **longDescription** ()

Returns the long description of this configuration.

*ConfigurationCategory* **category** ()

Returns the category of this configuration (mainly for UI structuring).

QString **changeHint** ()

Returns the change hint text of this configuration.

QVariant **defaultValue** ()

Returns the default value of this configuration.

*ConfigurationValueType* \***valueType** ()

Returns the value type of this configuration.

## Public Static Functions

*ConfigurationValueType* \***valueTypeString** ()

Returns the string type for configuration values.

*ConfigurationValueType* \***valueTypeInteger** ()

Returns the integer type for configuration values.

*ConfigurationValueType* \***valueTypeFloat** ()

Returns the floating point number type for configuration values.

*ConfigurationValueType* \***valueTypeBoolean** ()

Returns the boolean type for configuration values.

*ConfigurationValueType* \***valueTypeLocalFilePath** ()

Returns the filepath type for configuration values.

## Private Members

*ConfigurationManager* \*mgr

**class ConfigurationValue**

*#include <configurationvalue.h>* Abstract base class for a configuration value which is managed in the *Tuning* dialog.

Each instance can get and set the value associated to it.

You should typically not need to override it. See *sh::configuration::ConfigurationManager*. See Shallot documentation for more details.

Subclassed by *sh::configuration::ConfigurationManager::ConfigurationValueImpl*

## Public Functions

**ConfigurationValue** (QString *name*, QVariant *deflt*, QString *desc*, QString *longdesc*, *ConfigurationCategory* *cat*, *ConfigurationValueType* \**valuetype*, QString *changehint*)

int **getConfigValueInt** ()

Returns the current value of this instance as integer.

double **getConfigValueFloat** ()

Returns the current value of this instance as floating point number.

bool **getConfigValueBool** ()

Returns the current value of this instance as boolean.

QString **getConfigValueString** ()

Returns the current value of this instance as string.

QVariant **getConfigValueVariant** () = 0

Returns the current value of this instance as variant.

void **setConfigValue** (QVariant *value*) = 0

Sets a new configuration value.

bool **mayChange** () = 0

Returns if changes are allowed.

void **consumeValue** (QObject \**o*, std::function<void>

> *h* Adds a consumer function, triggered directly and whenever the value changes.

### Parameters

- *o*: For lifetime monitoring.

QString **name** ()

Returns the internal name of this configuration.

QString **description** ()

Returns the short description of this configuration.

QString **longDescription** ()

Returns the long description of this configuration.

*ConfigurationCategory* **category** ()

Returns the category of this configuration (mainly for UI structuring).

QString **changeHint** ()

Returns the change hint text of this configuration.

QVariant **defaultValue** ()

Returns the default value of this configuration.

*ConfigurationValueType* \***valueType** ()

Returns the value type of this configuration.

## Public Static Functions

*ConfigurationValueType* \***valueTypeString** ()  
Returns the string type for configuration values.

*ConfigurationValueType* \***valueTypeInteger** ()  
Returns the integer type for configuration values.

*ConfigurationValueType* \***valueTypeFloat** ()  
Returns the floating point number type for configuration values.

*ConfigurationValueType* \***valueTypeBoolean** ()  
Returns the boolean type for configuration values.

*ConfigurationValueType* \***valueTypeLocalFilePath** ()  
Returns the filepath type for configuration values.

## Friends

**friend class** ConfigurationManager

**friend class** ConfigurationValueType

**class** ConfigurationValueType  
#include <configurationvaluetype.h> Abstract base class for a configuration value type.  
Each subclass implements support for a certain type of configuration values (e.g. string, integer, ...).  
Subclassed by *sh::configuration::ConfigurationValueTypeBoolean*,  
*sh::configuration::ConfigurationValueTypeFloat*, *sh::configuration::ConfigurationValueTypeInteger*,  
*sh::configuration::ConfigurationValueTypeLocalFilePath*, *sh::configuration::ConfigurationValueTypeString*

## Public Functions

QString **valueDescription** (QVariant v)  
QVariant **parse** (QString input)

**class** ConfigurationValueTypeBoolean : public *sh::configuration::ConfigurationValueType*  
#include <configurationvaluetype.h> Configuration value type 'boolean'.

## Public Functions

QString **valueDescription** (QVariant v)  
QVariant **parse** (QString input)

**class** ConfigurationValueTypeFloat : public *sh::configuration::ConfigurationValueType*  
#include <configurationvaluetype.h> Configuration value type 'float'.

### Public Functions

QString **valueDescription** (QVariant *v*)

QVariant **parse** (QString *input*)

**class ConfigurationValueTypeInteger : public** *sh::configuration::ConfigurationValueType*  
*#include <configurationvaluetype.h>* Configuration value type ‘integer’.

### Public Functions

QString **valueDescription** (QVariant *v*)

QVariant **parse** (QString *input*)

**class ConfigurationValueTypeLocalFilePath : public** *sh::configuration::ConfigurationValueType*  
*#include <configurationvaluetype.h>* Configuration value type ‘file path’.

### Public Functions

QString **valueDescription** (QVariant *v*)

QVariant **parse** (QString *input*)

**class ConfigurationValueTypeString : public** *sh::configuration::ConfigurationValueType*  
*#include <configurationvaluetype.h>* Configuration value type ‘string’.

### Public Functions

QString **valueDescription** (QVariant *v*)

QVariant **parse** (QString *input*)

### namespace detailcolumns

Implementations of detail columns.

Subclasses of *sh::filesystem::DetailColumn* (and possibly some auxiliary stuff). They are shown in the fileviews and can be queried in code.

**class DetailColumnCustomAttributes : public** *sh::filesystem::DetailColumn*, **public** *sh::base::Singleton*  
*#include <detailcolumncustomattributes.h>* Detail column which determines the custom attributes.

Custom attributes are filesystem handler specific values about files (like permissions).

### Public Functions

void **applyValueByEurl** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*,  
*sh::filesystem::Operation* \**op*, QVariant *value*) **override**

QString **name** ()

The internal unique name.

QString **displayName** ()

The display name.

bool **isValueAvailable** (std::shared\_ptr<FilesystemNode> *node*)

Checks if a value for this detail is available for one given node.

QVariant **value** (std::shared\_ptr<FilesystemNode> *node*, bool *ignoreAged* = false)

Returns the value for this detail for one given node.

**Parameters**

- *ignoreAged*: If no value should be returned which is older than the last request (not useful for nearly all cases).

QString **displayValue** (std::shared\_ptr<FilesystemNode> *node*, **const** *sh::filesystem::FilesystemModelFileviewProxy* \**viewmodel*)

Returns the stringified value for this details for one given node considering the configuration of a view.

QVariant **requestValue** (std::shared\_ptr<FilesystemNode> *node*, *sh::filesystem::Operation* \**op* = 0, bool *withNodeValues* = false, bool *withOperationsCache* = true)

Requests to determine the value for this detail for one given node. Blocks until the value is available. Get it with the value method afterwards.

bool **sort\_doTypediff** ()

If sorting should separate files and directories.

int **sort\_order** (std::shared\_ptr<FilesystemNode> *n1*, std::shared\_ptr<FilesystemNode> *n2*)

The sorting order.

uint **displayIndex** ()

Controls which place this column should get in the user interface.

QVariant **computeValue** (std::shared\_ptr<FilesystemNode> *node*, *sh::filesystem::Operation* \**op*)

Returns a freshly determined value for this column and the given node. It neither asks nor populates any caches or storages.

int **defaultWidth** ()

bool **isRightAligned** ()

void **applyValueByEurl** (std::shared\_ptr<**const** *sh::filesystem::Eurl*> *eurl*, *sh::filesystem::Operation* \**op*, QVariant *value*)

Applies the detail value to a given eurl (without using any caches).

Used for transferring some details when a file transfer occurs.

Optionally implement the one of the `applyValueBy...` methods and register the instance with *sh::filesystem::FilesystemOperation::addTransferrableDetailColumn*. For performance reasons, you should decide to implement `applyValueByEurl`.

void **applyValueByNode** (std::shared\_ptr<FilesystemNode> *node*, *sh::filesystem::Operation* \**op*, QVariant *value*)

Applies the detail value to a given node (without using any caches).

Used for transferring some details when a file transfer occurs.

Optionally implement the one of the `applyValueBy...` methods and register the instance with *sh::filesystem::FilesystemOperation::addTransferrableDetailColumn*. For performance reasons, you should decide to implement `applyValueByEurl`.

void **doShutdown** ()

Executes singleton shutdown.

void **shutdown** ()

Shutdown down this singleton.

bool **isAlive** ()

Returns if this singleton is alive (true until its shutdown begins).



## Public Static Functions

QMap<QString, QString> **getMapByValue** (QVariant *val*)

Converts the QVariant(QString) from native detail column representation to a QString/QString map.

QVariant **VALUE\_UNAVAILABLE** ()

A special value expressing that the value is not yet available.

void **registerKnownDetailColumn** (QString *name*, std::shared\_ptr<DetailColumn> *column*)

std::shared\_ptr<DetailColumn> **findKnownDetailColumn** (QString *name*)

## Public Static Attributes

const uint **DISPLAYINDEX\_CORE** = 10000

const uint **DISPLAYINDEX\_INTERESTING** = 20000

const uint **DISPLAYINDEX\_EXOTIC** = 30000

## Private Functions

DetailColumnCustomAttributes ()

**class** **DetailColumnDirectSymlinkTarget** : **public** *sh::filesystem::DetailColumn*, **public** *sh::base::Singleton*  
*#include <detailcolumndirectsymlinktarget.h>* Detail column which determines the direct symlink target (by resolving a link once).

## Public Functions

QString **name** ()

The internal unique name.

QString **displayName** ()

The display name.

bool **isValueAvailable** (std::shared\_ptr<FilesystemNode> *node*)

Checks if a value for this detail is available for one given node.

QVariant **value** (std::shared\_ptr<FilesystemNode> *node*, bool *ignoreAged* = false)

Returns the value for this detail for one given node.

### Parameters

- *ignoreAged*: If no value should be returned which is older than the last request (not useful for nearly all cases).

QString **displayValue** (std::shared\_ptr<FilesystemNode> *node*, **const** *sh::filesystem::FilesystemModelFileviewProxy* \**viewmodel*)

Returns the stringified value for this details for one given node considering the configuration of a view.

QVariant **requestValue** (std::shared\_ptr<FilesystemNode> *node*, *sh::filesystem::Operation* \**op* = 0, bool *withNodeValues* = false, bool *withOperationsCache* = true)

Requests to determine the value for this detail for one given node. Blocks until the value is available. Get it with the value method afterwards.

bool **sort\_doTypediff** ()

If sorting should separate files and directories.

int **sort\_order** (std::shared\_ptr<FilesystemNode> *n1*, std::shared\_ptr<FilesystemNode> *n2*)

The sorting order.

uint **displayIndex** ()

Controls which place this column should get in the user interface.

QVariant **computeValue** (std::shared\_ptr<FilesystemNode> *node*, *sh::filesystem::Operation*  
\**op*)

Returns a freshly determined value for this column and the given node. It neither asks nor populates any caches or storages.

int **defaultWidth** ()

bool **isRightAligned** ()

void **applyValueByEurl** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*,  
*sh::filesystem::Operation* \**op*, QVariant *value*)

Applies the detail value to a given eurl (without using any caches).

Used for transferring some details when a file transfer occurs.

Optionally implement the one of the applyValueBy... methods and register the instance with *sh::filesystem::FilesystemOperation::addTransferrableDetailColumn*. For performance reasons, you should decide to implement applyValueByEurl.

void **applyValueByNode** (std::shared\_ptr<FilesystemNode> *node*, *sh::filesystem::Operation*  
\**op*, QVariant *value*)

Applies the detail value to a given node (without using any caches).

Used for transferring some details when a file transfer occurs.

Optionally implement the one of the applyValueBy... methods and register the instance with *sh::filesystem::FilesystemOperation::addTransferrableDetailColumn*. For performance reasons, you should decide to implement applyValueByEurl.

void **doShutdown** ()

Executes singleton shutdown.

void **shutdown** ()

Shutdown down this singleton.

bool **isAlive** ()

Returns if this singleton is alive (true until its shutdown begins).

## Public Static Functions

QVariant **VALUE\_UNAVAILABLE** ()

A special value expressing that the value is not yet available.

void **registerKnownDetailColumn** (QString *name*, std::shared\_ptr<DetailColumn> *column*)

std::shared\_ptr<DetailColumn> **findKnownDetailColumn** (QString *name*)

## Public Static Attributes

```
const uint DISPLAYINDEX_CORE = 10000
const uint DISPLAYINDEX_INTERESTING = 20000
const uint DISPLAYINDEX_EXOTIC = 30000
```

## Private Functions

```
DetailColumnDirectSymlinkTarget ()
```

```
class DetailColumnExtendedAttributes : public sh::filesystem::DetailColumn, public sh::base::Singleton
#include <detailcolumnextendedattributes.h> Detail column which determines the extended at-
tributes.
```

Extended attributes are a feature of some filesystems.

## Public Functions

```
void applyValueByEurl (std::shared_ptr<const sh::filesystem::Eurl> eurl,
sh::filesystem::Operation *op, QVariant value) override
```

```
QString name ()
The internal unique name.
```

```
QString displayName ()
The display name.
```

```
bool isValueAvailable (std::shared_ptr<FilesystemNode> node)
Checks if a value for this detail is available for one given node.
```

```
QVariant value (std::shared_ptr<FilesystemNode> node, bool ignoreAged = false)
Returns the value for this detail for one given node.
```

### Parameters

- ignoreAged: If no value should be returned which is older than the last request (not useful for nearly all cases).

```
QString displayValue (std::shared_ptr<FilesystemNode> node, const
sh::filesystem::FilesystemModelFileviewProxy *viewmodel)
Returns the stringified value for this details for one given node considering the configuration of a
view.
```

```
QVariant requestValue (std::shared_ptr<FilesystemNode> node, sh::filesystem::Operation
*op = 0, bool withNodeValues = false, bool withOperationsCache
= true)
Requests to determine the value for this detail for one given node. Blocks until the value is
available. Get it with the value method afterwards.
```

```
bool sort_doTypediff ()
If sorting should separate files and directories.
```

```
int sort_order (std::shared_ptr<FilesystemNode> n1, std::shared_ptr<FilesystemNode> n2)
The sorting order.
```

```
uint displayIndex ()
Controls which place this column should get in the user interface.
```

QVariant **computeValue** (std::shared\_ptr<FilesystemNode> *node*, *sh::filesystem::Operation* \**op*)

Returns a freshly determined value for this column and the given node. It neither asks nor populates any caches or storages.

int **defaultWidth** ()

bool **isRightAligned** ()

void **applyValueByEurl** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*, *sh::filesystem::Operation* \**op*, QVariant *value*)

Applies the detail value to a given eurl (without using any caches).

Used for transferring some details when a file transfer occurs.

Optionally implement the one of the applyValueBy... methods and register the instance with *sh::filesystem::FilesystemOperation::addTransferrableDetailColumn*. For performance reasons, you should decide to implement applyValueByEurl.

void **applyValueByNode** (std::shared\_ptr<FilesystemNode> *node*, *sh::filesystem::Operation* \**op*, QVariant *value*)

Applies the detail value to a given node (without using any caches).

Used for transferring some details when a file transfer occurs.

Optionally implement the one of the applyValueBy... methods and register the instance with *sh::filesystem::FilesystemOperation::addTransferrableDetailColumn*. For performance reasons, you should decide to implement applyValueByEurl.

void **doShutdown** ()

Executes singleton shutdown.

void **shutdown** ()

Shutdown down this singleton.

bool **isAlive** ()

Returns if this singleton is alive (true until its shutdown begins).

## Public Static Functions

QMap<QString, QByteArray> **getMapByValue** (QVariant *val*)

Converts the QVariant(QString) from native detail column representation to a QString/QString map.

QVariant **VALUE\_UNAVAILABLE** ()

A special value expressing that the value is not yet available.

void **registerKnownDetailColumn** (QString *name*, std::shared\_ptr<DetailColumn> *column*)

std::shared\_ptr<DetailColumn> **findKnownDetailColumn** (QString *name*)

## Public Static Attributes

**const** uint **DISPLAYINDEX\_CORE** = 10000

**const** uint **DISPLAYINDEX\_INTERESTING** = 20000

**const** uint **DISPLAYINDEX\_EXOTIC** = 30000

## Private Functions

**DetailColumnExtendedAttributes** ()

**class** **DetailColumnFilesize** : **public** *sh::filesystem::DetailColumn*, **public** *sh::base::Singleton*  
*#include <detailcolumnfilesize.h>* Detail column which determines the file size.

## Public Functions

QString **name** ()

The internal unique name.

QString **displayName** ()

The display name.

bool **isValueAvailable** (std::shared\_ptr<FilesystemNode> *node*)

Checks if a value for this detail is available for one given node.

QVariant **value** (std::shared\_ptr<FilesystemNode> *node*, bool *ignoreAged* = false)

Returns the value for this detail for one given node.

### Parameters

- *ignoreAged*: If no value should be returned which is older than the last request (not useful for nearly all cases).

bool **isVisible** ()

If this detail shall be a visible column in the view.

QVariant **requestValue** (std::shared\_ptr<FilesystemNode> *node*, *sh::filesystem::Operation* \**op* = 0, bool *withNodeValues* = false, bool *withOperationsCache* = true)

Requests to determine the value for this detail for one given node. Blocks until the value is available. Get it with the value method afterwards.

bool **sort\_doTypediff** ()

If sorting should separate files and directories.

int **sort\_order** (std::shared\_ptr<FilesystemNode> *n1*, std::shared\_ptr<FilesystemNode> *n2*)

The sorting order.

uint **displayIndex** ()

Controls which place this column should get in the user interface.

QVariant **computeValue** (std::shared\_ptr<FilesystemNode> *node*, *sh::filesystem::Operation* \**op*)

Returns a freshly determined value for this column and the given node. It neither asks nor populates any caches or storages.

int **defaultWidth** ()

bool **isRightAligned** ()

```
void applyValueByEurl (std::shared_ptr<const sh::filesystem::Eurl> eurl,  
                      sh::filesystem::Operation *op, QVariant value)
```

Applies the detail value to a given eurl (without using any caches).

Used for transferring some details when a file transfer occurs.

Optionally implement the one of the `applyValueBy...` methods and register the instance with *sh::filesystem::FilesystemOperation::addTransferrableDetailColumn*. For performance reasons, you should decide to implement `applyValueByEurl`.

```
void applyValueByNode (std::shared_ptr<FilesystemNode> node, sh::filesystem::Operation  
                      *op, QVariant value)
```

Applies the detail value to a given node (without using any caches).

Used for transferring some details when a file transfer occurs.

Optionally implement the one of the `applyValueBy...` methods and register the instance with *sh::filesystem::FilesystemOperation::addTransferrableDetailColumn*. For performance reasons, you should decide to implement `applyValueByEurl`.

```
void doShutdown ()
```

Executes singleton shutdown.

```
void shutdown ()
```

Shutdown down this singleton.

```
bool isAlive ()
```

Returns if this singleton is alive (true until its shutdown begins).

## Public Static Functions

```
QVariant VALUE_UNAVAILABLE ()
```

A special value expressing that the value is not yet available.

```
void registerKnownDetailColumn (QString name, std::shared_ptr<DetailColumn> col-  
                               umn)
```

```
std::shared_ptr<DetailColumn> findKnownDetailColumn (QString name)
```

## Public Static Attributes

```
const uint DISPLAYINDEX_CORE = 10000
```

```
const uint DISPLAYINDEX_INTERESTING = 20000
```

```
const uint DISPLAYINDEX_EXOTIC = 30000
```

## Private Functions

```
DetailColumnFileSize ()
```

```
class DetailColumnMimetype : public sh::filesystem::DetailColumn, public sh::base::Singleton  
    #include <detailcolumnmimetype.h> Detail column which determines the file's mimetype.
```

## Public Functions

QString **name** ()

The internal unique name.

QString **displayName** ()

The display name.

bool **isValueAvailable** (std::shared\_ptr<FilesystemNode> *node*)

Checks if a value for this detail is available for one given node.

QVariant **value** (std::shared\_ptr<FilesystemNode> *node*, bool *ignoreAged* = false)

Returns the value for this detail for one given node.

### Parameters

- *ignoreAged*: If no value should be returned which is older than the last request (not useful for nearly all cases).

bool **isVisible** ()

If this detail shall be a visible column in the view.

QVariant **requestValue** (std::shared\_ptr<FilesystemNode> *node*, *sh::filesystem::Operation* \**op* = 0, bool *withNodeValues* = false, bool *withOperationsCache* = true)

Requests to determine the value for this detail for one given node. Blocks until the value is available. Get it with the value method afterwards.

bool **sort\_doTypediff** ()

If sorting should separate files and directories.

int **sort\_order** (std::shared\_ptr<FilesystemNode> *n1*, std::shared\_ptr<FilesystemNode> *n2*)

The sorting order.

uint **displayIndex** ()

Controls which place this column should get in the user interface.

QVariant **computeValue** (std::shared\_ptr<FilesystemNode> *node*, *sh::filesystem::Operation* \**op*)

Returns a freshly determined value for this column and the given node. It neither asks nor populates any caches or storages.

int **defaultWidth** ()

bool **isRightAligned** ()

void **applyValueByEurl** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*, *sh::filesystem::Operation* \**op*, QVariant *value*)

Applies the detail value to a given eurl (without using any caches).

Used for transferring some details when a file transfer occurs.

Optionally implement the one of the applyValueBy... methods and register the instance with *sh::filesystem::FilesystemOperation::addTransferrableDetailColumn*. For performance reasons, you should decide to implement applyValueByEurl.

void **applyValueByNode** (std::shared\_ptr<FilesystemNode> *node*, *sh::filesystem::Operation* \**op*, QVariant *value*)

Applies the detail value to a given node (without using any caches).

Used for transferring some details when a file transfer occurs.

Optionally implement the one of the applyValueBy... methods and register the instance with *sh::filesystem::FilesystemOperation::addTransferrableDetailColumn*. For performance reasons, you should decide to implement applyValueByEurl.

void **doShutdown** ()  
Executes singleton shutdown.

void **shutdown** ()  
Shutdown down this singleton.

bool **isAlive** ()  
Returns if this singleton is alive (true until its shutdown begins).

### Public Static Functions

QVariant **VALUE\_UNAVAILABLE** ()  
A special value expressing that the value is not yet available.

void **registerKnownDetailColumn** (QString *name*, std::shared\_ptr<DetailColumn> *column*)

std::shared\_ptr<DetailColumn> **findKnownDetailColumn** (QString *name*)

### Public Static Attributes

const uint **DISPLAYINDEX\_CORE** = 10000

const uint **DISPLAYINDEX\_INTERESTING** = 20000

const uint **DISPLAYINDEX\_EXOTIC** = 30000

### Private Functions

**DetailColumnMimetype** ()

**class DetailColumnMtime : public *sh::filesystem::DetailColumn*, public *sh::base::Singleton***  
*#include <detailcolumnmtime.h>* Detail column which determines the file's modification time.

### Public Functions

void **applyValueByEurl** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*,  
*sh::filesystem::Operation* \**op*, QVariant *value*) **override**

QString **name** ()  
The internal unique name.

QString **displayName** ()  
The display name.

bool **isValueAvailable** (std::shared\_ptr<FilesystemNode> *node*)  
Checks if a value for this detail is available for one given node.

QVariant **value** (std::shared\_ptr<FilesystemNode> *node*, bool *ignoreAged* = false)  
Returns the value for this detail for one given node.

**Parameters**

- *ignoreAged*: If no value should be returned which is older than the last request (not useful for nearly all cases).

bool **isVisible** ()  
If this detail shall be a visible column in the view.



QVariant **requestValue** (std::shared\_ptr<FilesystemNode> *node*, *sh::filesystem::Operation* \**op* = 0, bool *withNodeValues* = false, bool *withOperationsCache* = true)  
 Requests to determine the value for this detail for one given node. Blocks until the value is available. Get it with the value method afterwards.

bool **sort\_doTypediff** ()  
 If sorting should separate files and directories.

int **sort\_order** (std::shared\_ptr<FilesystemNode> *n1*, std::shared\_ptr<FilesystemNode> *n2*)  
 The sorting order.

uint **displayIndex** ()  
 Controls which place this column should get in the user interface.

QVariant **computeValue** (std::shared\_ptr<FilesystemNode> *node*, *sh::filesystem::Operation* \**op*)  
 Returns a freshly determined value for this column and the given node. It neither asks nor populates any caches or storages.

int **defaultWidth** ()

bool **isRightAligned** ()

void **applyValueByEurl** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*, *sh::filesystem::Operation* \**op*, QVariant *value*)  
 Applies the detail value to a given eurl (without using any caches).  
 Used for transferring some details when a file transfer occurs.  
 Optionally implement the one of the applyValueBy... methods and register the instance with *sh::filesystem::FilesystemOperation::addTransferrableDetailColumn*. For performance reasons, you should decide to implement applyValueByEurl.

void **applyValueByNode** (std::shared\_ptr<FilesystemNode> *node*, *sh::filesystem::Operation* \**op*, QVariant *value*)  
 Applies the detail value to a given node (without using any caches).  
 Used for transferring some details when a file transfer occurs.  
 Optionally implement the one of the applyValueBy... methods and register the instance with *sh::filesystem::FilesystemOperation::addTransferrableDetailColumn*. For performance reasons, you should decide to implement applyValueByEurl.

void **doShutdown** ()  
 Executes singleton shutdown.

void **shutdown** ()  
 Shutdown down this singleton.

bool **isAlive** ()  
 Returns if this singleton is alive (true until its shutdown begins).

## Public Static Functions

QVariant **VALUE\_UNAVAILABLE** ()

A special value expressing that the value is not yet available.

void **registerKnownDetailColumn** (QString *name*, std::shared\_ptr<DetailColumn> *column*)

std::shared\_ptr<DetailColumn> **findKnownDetailColumn** (QString *name*)

## Public Static Attributes

const uint **DISPLAYINDEX\_CORE** = 10000

const uint **DISPLAYINDEX\_INTERESTING** = 20000

const uint **DISPLAYINDEX\_EXOTIC** = 30000

## Private Functions

**DetailColumnMtime** ()

**class DetailColumnResolvedSymlink : public *sh::filesystem::DetailColumn*, public *sh::base::Singleton***  
*#include <detailcolumnresolvedsymlink.h>* Detail column which determines the symlink target (by resolving links recursively).

This detail column has some active behavior. It informs the filesystem model about the ‘buddy nodes’.

## Public Functions

QString **name** ()

The internal unique name.

QString **displayName** ()

The display name.

bool **isValueAvailable** (std::shared\_ptr<FilesystemNode> *node*)

Checks if a value for this detail is available for one given node.

QVariant **value** (std::shared\_ptr<FilesystemNode> *node*, bool *ignoreAged* = false)

Returns the value for this detail for one given node.

### Parameters

- *ignoreAged*: If no value should be returned which is older than the last request (not useful for nearly all cases).

QVariant **requestValue** (std::shared\_ptr<FilesystemNode> *node*, *sh::filesystem::Operation* \**op* = 0, bool *withNodeValues* = false, bool *withOperationsCache* = true)

Requests to determine the value for this detail for one given node. Blocks until the value is available. Get it with the value method afterwards.

bool **sort\_doTypediff** ()

If sorting should separate files and directories.

int **sort\_order** (std::shared\_ptr<FilesystemNode> *n1*, std::shared\_ptr<FilesystemNode> *n2*)

The sorting order.

uint **displayIndex** ()

Controls which place this column should get in the user interface.

QVariant **computeValue** (std::shared\_ptr<FilesystemNode> *node*, *sh::filesystem::Operation* \**op*)

Returns a freshly determined value for this column and the given node. It neither asks nor populates any caches or storages.

int **defaultWidth** ()

bool **isRightAligned** ()

void **applyValueByEurl** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*, *sh::filesystem::Operation* \**op*, QVariant *value*)

Applies the detail value to a given eurl (without using any caches).

Used for transferring some details when a file transfer occurs.

Optionally implement the one of the applyValueBy... methods and register the instance with *sh::filesystem::FilesystemOperation::addTransferrableDetailColumn*. For performance reasons, you should decide to implement applyValueByEurl.

void **applyValueByNode** (std::shared\_ptr<FilesystemNode> *node*, *sh::filesystem::Operation* \**op*, QVariant *value*)

Applies the detail value to a given node (without using any caches).

Used for transferring some details when a file transfer occurs.

Optionally implement the one of the applyValueBy... methods and register the instance with *sh::filesystem::FilesystemOperation::addTransferrableDetailColumn*. For performance reasons, you should decide to implement applyValueByEurl.

void **doShutdown** ()

Executes singleton shutdown.

void **shutdown** ()

Shutdown down this singleton.

bool **isAlive** ()

Returns if this singleton is alive (true until its shutdown begins).

## Public Static Functions

QVariant **VALUE\_UNAVAILABLE** ()

A special value expressing that the value is not yet available.

void **registerKnownDetailColumn** (QString *name*, std::shared\_ptr<DetailColumn> *column*)

std::shared\_ptr<DetailColumn> **findKnownDetailColumn** (QString *name*)

### Public Static Attributes

```
const uint DISPLAYINDEX_CORE = 10000
const uint DISPLAYINDEX_INTERESTING = 20000
const uint DISPLAYINDEX_EXOTIC = 30000
```

### Private Functions

```
DetailColumnResolvedSymlink()
```

### namespace exceptions

Exceptions.

The class hierarchy of Shallot exceptions as well as some utilities for exception handling.

### Enums

#### enum ExecuteGuardFlag

*Values:*

```
enumerator MANUAL_RETRY_ENABLED = 1 << 0
enumerator AUTOMATIC_RETRY_ENABLED = 1 << 1
enumerator LOGGING_DISABLED = 1 << 2
enumerator RESUMEABLE_PROGRAM_ERROR_END_WITH_USER_FEEDBACK_HERE = 1 << 3
enumerator CANCELABLE_UP_TO_HERE = 1 << 4
enumerator ALL_ERRORS_KILL_SHALLOT = 1 << 5
enumerator IGNORE_ALL_RESUMEABLE_ERRORS_SILENTLY = 1 << 6
```

#### class ArgumentException: public *sh::exceptions::ProgramException*

*#include <argumentexception.h>* Shallot exception for failed operation due to invalid arguments given to some program part. It typically allows resume but not retry (special cases may override each of them).

Subclassed by *sh::filesystem::EurlMisformattedException*

### Public Functions

```
ArgumentException(ExceptionData data)
```

```
QString message() const
```

```
QString name() const
```

```
QString classes() const
```

```
QString details() const
```

```
QString callstack() const
```

```
QString auxiliary() const
```

```
QString customValue(QString key) const
```

```
bool isRuntimeException() const
```

```

bool isProgramException () const
bool isRetryable () const
bool isResumeable () const
bool isDetailsAreInteresting () const
int autoRetryRecommendedIn () const
void setResumeable (bool v)
void setReTryable (bool v)
void setCustomValue (QString key, QString value)
bool isClass (QString classname)
sh::exceptions::ExceptionData data ()

```

### Public Static Functions

```

template<class Handler>
std::shared_ptr<RegisterHandler<Handler>> createRegisterHandler (Handler handler,
                                                                QS-
                                                                tack<Handler>
                                                                *stack)

void executeGuarded (std::function<void>
    > fct) int flags = 0 Executes some code with some standard exection handling around.

Parameters
    • fct: The code to execute guarded.
    • flags: Flags of ExecuteGuardFlag for choosing the behavior.

void executeGuarded_errorpanel (std::function<void>
    > fct) int flags = 0

QString _demangle (QString l)

void exceptionDialog (sh::exceptions::Exception &e, bool *doRetry)

```

### Public Static Attributes

```

const QString UNDEFINED_ERROR_OCCURRED = QObject::tr("An unspecified error occurred.")
const QString SHALLOT_MUST_CLOSE_TEXT = QObject::tr("Your Shallot process is disturbed by an error and ne
const QString Value_isShallotException = "_isShallotException"

```

### class CancelException

*#include <exception.h>* A special exception for cancellation of some action on user behalf.

A very special exception outside of the hierarchy. It is only used by the infrastructure; never throw it directly.

## Public Functions

**CancelException()**

**class Exception**

*#include <exception.h>* Shallot exception base class. Also contains some static methods for general work with exceptions.

Subclassed by *sh::exceptions::ProgramException*, *sh::exceptions::RuntimeException*, *sh::scripting::ScriptingEngine::ScriptedException*

## Public Functions

QString **message()** const

QString **name()** const

QString **classes()** const

QString **details()** const

QString **callstack()** const

QString **auxiliary()** const

QString **customValue**(QString *key*) const

bool **isRuntimeException()** const

bool **isProgramException()** const

bool **isRetryable()** const

bool **isResumeable()** const

bool **isDetailsAreInteresting()** const

int **autoRetryRecommendedIn()** const

void **setResumeable**(bool *v*)

void **setRetryable**(bool *v*)

void **setCustomValue**(QString *key*, QString *value*)

bool **isClass**(QString *classname*)

**Exception**(*ExceptionData* *data*)

**Exception**()

*sh::exceptions::ExceptionData* **data**()

## Public Static Functions

template<class **Handler**>

std::shared\_ptr<*RegisterHandler*<*Handler*>> **createRegisterHandler**(*Handler* *handler*,  
 QS-  
 tack<*Handler*>  
 \**stack*)

void **executeGuarded**(std::function<void>)

> *fcntl* flags = 0 Executes some code with some standard exection handling around.

**Parameters**

- `fct`: The code to execute guarded.
- `flags`: Flags of *ExecuteGuardFlag* for choosing the behavior.

```
void executeGuarded_errorpanel (std::function<void>
    > fct, int flags = 0)
```

```
QString _demangle (QString l)
```

```
void exceptionDialog (sh::exceptions::Exception &e, bool *doRetry)
```

**Public Static Attributes**

```
const QString UNDEFINED_ERROR_OCCURRED = QObject::tr("An unspecified error occurred.")
```

```
const QString SHALLOT_MUST_CLOSE_TEXT = QObject::tr("Your Shallot process is disturbed by an error and ne
```

```
const QString Value_isShallotException = "_isShallotException"
```

**Private Static Functions**

```
void exceptionDialog (QString error1, QString error2, QString details, QString icon, bool
    mayRetry, bool mayClose, bool mayCancel, bool showLoglabelAnd-
    Details, bool *doRetry)
```

**Private Static Attributes**

```
bool _inShutdown = false
```

```
QMutex _inShutdownMutex
```

```
class HandlerSettings
    #include <exception.h>
```

**Public Members**

```
QString cancelText
```

```
QStack<std::function<bool (sh::exceptions::Exception&) >> exceptionHandler_retryable
```

```
QStack<std::function<void (sh::exceptions::Exception&) >> exceptionHandler_resumeable
```

```
QStack<std::function<void (sh::exceptions::Exception&) >> exceptionHandler_hard
```

```
QStack<std::function<void () >> exceptionHandler_cancel
```

**Private Functions**

```
HandlerSettings ()
```

### Friends

```
friend class ExceptionHandlerSettingsManager  
template<class Handler>  
class RegisterHandler  
    #include <exception.h>
```

### Public Functions

```
RegisterHandler (Handler handler, QStack<Handler> *stack)  
~RegisterHandler ()
```

### Private Members

```
QStack<Handler> *_stack  
class ExceptionData : public QMap<QString, QString>  
    #include <exception.h> Used for specifying metadata for a sh::exceptions::Exception.
```

### Public Functions

```
ExceptionData (const ExceptionData &o)  
ExceptionData (const QMap<QString, QString> &o)  
ExceptionData ()  
ExceptionData name (QString name)  
ExceptionData classes (QString classes)  
ExceptionData aux (QString aux)  
ExceptionData details (QString details)  
ExceptionData message (QString message)  
ExceptionData runtime ()  
ExceptionData program ()  
ExceptionData retryable (bool v = true)  
ExceptionData resumeable (bool v = true)  
ExceptionData autoRetryRecommended (int v = -1)  
ExceptionData detailsAreInteresting (bool v = true)  
class ExceptionHandlerSettingsManager : public sh::base::Singleton  
    #include <exception.h> Managing how to default-handle unhandled exceptions.
```



## Public Functions

*Exception::HandlerSettings* \***handlerSettings** ()  
Returns the current *Exception::HandlerSettings*.

**~ExceptionHandlerSettingsManager** ()

void **doInitialize** ()  
Executes singleton initialization.

void **doShutdown** ()  
Executes singleton shutdown.

void **shutdown** ()  
Shutdown down this singleton.

bool **isAlive** ()  
Returns if this singleton is alive (true until its shutdown begins).

## Private Functions

**ExceptionHandlerSettingsManager** ()

## Private Members

QMap<QThread\*, *Exception::HandlerSettings*\*> **\_handlerSettings**

QMutex **\_mutex**

**class IOException**: public *sh::exceptions::RuntimeException*  
*#include <ioexception.h>* Shallot exception in IO. It allows resume and typically allows retry (special cases may override each of them).  
 Subclassed by *sh::exceptions::PermissionDeniedException*, *sh::filesystem::Operation::MaxAllowedSizeRatioPerPartEx*

## Public Functions

**IOException** (*ExceptionData* data)

QString **message** () const

QString **name** () const

QString **classes** () const

QString **details** () const

QString **callstack** () const

QString **auxiliary** () const

QString **customValue** (QString key) const

bool **isRuntimeException** () const

bool **isProgramException** () const

bool **isRetryable** () const

bool **isResumeable** () const

bool **isDetailsAreInteresting** () const

```
int autoRetryRecommendedIn () const
void setResumeable (bool v)
void setReTryable (bool v)
void setCustomValue (QString key, QString value)
bool isClass (QString classname)
sh::exceptions::ExceptionData data ()
```

## Public Static Functions

```
template<class Handler>
std::shared_ptr<RegisterHandler<Handler>> createRegisterHandler (Handler    han-
                                                                dler,    QS-
                                                                tack<Handler>
                                                                *stack)

void executeGuarded (std::function<void>)
    > fc int flags = 0 Executes some code with some standard exection handling around.

Parameters
    • fc t: The code to execute guarded.
    • flags: Flags of ExecuteGuardFlag for choosing the behavior.

void executeGuarded_errorpanel (std::function<void>)
    > fc int flags = 0

QString _demangle (QString l)

void exceptionDialog (sh::exceptions::Exception &e, bool *doRetry)
```

## Public Static Attributes

```
const QString UNDEFINED_ERROR_OCCURRED = QObject::tr("An unspecified error occurred.")
const QString SHALLOT_MUST_CLOSE_TEXT = QObject::tr("Your Shallot process is disturbed by an error and ne
const QString Value_isShallotException = "_isShallotException"

class PermissionDeniedException: public sh::exceptions::IOException
    #include <permissiondeniedexception.h> Shallot exception for something forbidden was tried to ex-
    ecute. It allows resume and retry (special cases may override each of them).
```

## Public Functions

```
PermissionDeniedException (ExceptionData data)

QString message () const
QString name () const
QString classes () const
QString details () const
QString callstack () const
QString auxiliary () const
```

```

QString customValue (QString key) const
bool isRuntimeException () const
bool isProgramException () const
bool isRetryable () const
bool isResumeable () const
bool isDetailsAreInteresting () const
int autoRetryRecommendedIn () const
void setResumeable (bool v)
void setReTryable (bool v)
void setCustomValue (QString key, QString value)
bool isClass (QString classname)
sh::exceptions::ExceptionData data ()

```

### Public Static Functions

```

template<class Handler>
std::shared_ptr<RegisterHandler<Handler>> createRegisterHandler (Handler handler,
                                                                QS-
                                                                tack<Handler>
                                                                *stack)

void executeGuarded (std::function<void>
    > fctint flags = 0) Executes some code with some standard exection handling around.

Parameters
    • fct: The code to execute guarded.
    • flags: Flags of ExecuteGuardFlag for choosing the behavior.

void executeGuarded_errorpanel (std::function<void>
    > fctint flags = 0)

QString _demangle (QString l)

void exceptionDialog (sh::exceptions::Exception &e, bool *doRetry)

```

### Public Static Attributes

```

const QString UNDEFINED_ERROR_OCCURRED = QObject::tr("An unspecified error occurred.")
const QString SHALLOT_MUST_CLOSE_TEXT = QObject::tr("Your Shallot process is disturbed by an error and ne
const QString Value_isShallotException = "_isShallotException"

class ProgramException: public sh::exceptions::Exception
    #include <programexception.h> Shallot exception for internal bugs in Shallot. It optionally allows
    resume but no retry (special cases may override each of them).

    Subclassed by sh::exceptions::ArgumentException, sh::exceptions::ThreadingException

```

## Public Functions

```
ProgramException (ExceptionData data)
QString message () const
QString name () const
QString classes () const
QString details () const
QString callstack () const
QString auxiliary () const
QString customValue (QString key) const
bool isRuntimeException () const
bool isProgramException () const
bool isRetryable () const
bool isResumeable () const
bool isDetailsAreInteresting () const
int autoRetryRecommendedIn () const
void setResumeable (bool v)
void setReTryable (bool v)
void setCustomValue (QString key, QString value)
bool isClass (QString classname)
sh::exceptions::ExceptionData data ()
```

## Public Static Functions

```
template<class Handler>
std::shared_ptr<RegisterHandler<Handler>> createRegisterHandler (Handler han-
                                                                    dler,
                                                                    QS-
                                                                    tack<Handler>
                                                                    *stack)

void executeGuarded (std::function<void>
    > fct int flags = 0) Executes some code with some standard exection handling around.

Parameters
    • fct: The code to execute guarded.
    • flags: Flags of ExecuteGuardFlag for choosing the behavior.

void executeGuarded_errorpanel (std::function<void>
    > fct int flags = 0)

QString _demangle (QString l)

void exceptionDialog (sh::exceptions::Exception &e, bool *doRetry)
```

## Public Static Attributes

**const** QString **UNDEFINED\_ERROR\_OCCURRED** = QObject::tr("An unspecified error occurred.")

**const** QString **SHALLOT\_MUST\_CLOSE\_TEXT** = QObject::tr("Your Shallot process is disturbed by an error and ne

**const** QString **Value\_isShallotException** = "\_isShallotException"

**class** **RuntimeException** : public *sh::exceptions::Exception*

*#include <runtimeexception.h>* Shallot exception for failed operation due to (often external) runtime effects. It allows resume and optionally allows retry (special cases may override each of them).

Subclassed by *sh::exceptions::IOException*

## Public Functions

**RuntimeException** (*ExceptionData* data)

QString **message** () **const**

QString **name** () **const**

QString **classes** () **const**

QString **details** () **const**

QString **callstack** () **const**

QString **auxiliary** () **const**

QString **customValue** (QString key) **const**

bool **isRuntimeException** () **const**

bool **isProgramException** () **const**

bool **isRetryable** () **const**

bool **isResumeable** () **const**

bool **isDetailsAreInteresting** () **const**

int **autoRetryRecommendedIn** () **const**

void **setResumeable** (bool v)

void **setReTryable** (bool v)

void **setCustomValue** (QString key, QString value)

bool **isClass** (QString classname)

*sh::exceptions::ExceptionData* **data** ()

## Public Static Functions

```
template<class Handler>
std::shared_ptr<RegisterHandler<Handler>> createRegisterHandler (Handler    han-
                                                                    dler,    QS-
                                                                    tack<Handler>
                                                                    *stack)
```

```
void executeGuarded (std::function<void>
    > fct; int flags = 0) Executes some code with some standard exection handling around.
```

### Parameters

- *fct*: The code to execute guarded.
- *flags*: Flags of *ExecuteGuardFlag* for choosing the behavior.

```
void executeGuarded_errorpanel (std::function<void>
    > fct; int flags = 0)
```

```
QString _demangle (QString l)
```

```
void exceptionDialog (sh::exceptions::Exception &e, bool *doRetry)
```

## Public Static Attributes

```
const QString UNDEFINED_ERROR_OCCURRED = QObject::tr("An unspecified error occurred.")
```

```
const QString SHALLOT_MUST_CLOSE_TEXT = QObject::tr("Your Shallot process is disturbed by an error and ne
```

```
const QString Value_isShallotException = "_isShallotException"
```

```
class ThreadAbortException
```

```
    #include <exception.h> Only used by the infrastructure; never throw it directly.
```

## Public Functions

```
ThreadAbortException ()
```

```
class ThreadingException: public sh::exceptions::ProgramException
```

```
    #include <threadingexception.h> Shallot exception for misuse of threading (e.g. wrong caller thread).  
    It does not allow resume (special cases may override each of them).
```

## Public Functions

```
ThreadingException (ExceptionData data)
```

```
QString message () const
```

```
QString name () const
```

```
QString classes () const
```

```
QString details () const
```

```
QString callstack () const
```

```
QString auxiliary () const
```

```
QString customValue (QString key) const
```

```
bool isRuntimeException () const
```

```

bool isProgramException () const
bool isRetryable () const
bool isResumeable () const
bool isDetailsAreInteresting () const
int autoRetryRecommendedIn () const
void setResumeable (bool v)
void setReTryable (bool v)
void setCustomValue (QString key, QString value)
bool isClass (QString classname)
sh::exceptions::ExceptionData data ()

```

### Public Static Functions

```

template<class Handler>
std::shared_ptr<RegisterHandler<Handler>> createRegisterHandler (Handler handler,
                                                                QS-
                                                                tack<Handler>
                                                                *stack)

void executeGuarded (std::function<void>
    > fct) int flags = 0 Executes some code with some standard exection handling around.

Parameters
    • fct: The code to execute guarded.
    • flags: Flags of ExecuteGuardFlag for choosing the behavior.

void executeGuarded_errorpanel (std::function<void>
    > fct) int flags = 0

QString _demangle (QString l)

void exceptionDialog (sh::exceptions::Exception &e, bool *doRetry)

```

### Public Static Attributes

```

const QString UNDEFINED_ERROR_OCCURRED = QObject::tr("An unspecified error occurred.")
const QString SHALLOT_MUST_CLOSE_TEXT = QObject::tr("Your Shallot process is disturbed by an error and ne
const QString Value_isShallotException = "_isShallotException"

```

### namespace filepropertydialogtabs

Implementations of tabs in a file property dialog.

Subclasses of *sh::ui::FilePropertyDialogTab* (and possibly some auxiliary stuff). They implement content in file property dialogs.

```

class FilePropertyDialogTabExtendedAttributes : public sh::ui::FilePropertyDialogTab
    #include <filepropertydialogtabextendedattributes.h> Properties dialog tab for extended attributes.

```

## Public Functions

**FilePropertyDialogTabExtendedAttributes** ()

QString **title** () **override**

The tab title. .

QList<QString> **properties** () **override**

Returns the list of property labels (one entry for each widget).

Each property is displayed with a header and a widget (created in `createWidget`) with some content (populated in `updateWidget`).

void **populateWidget** (int *i*, *sh::ui::FilePropertyDialogTabActionsView* \**widget*) **override**

Populates the tab widget for the *i*-th property.

Typically uses the `FilePropertyDialog::create*` methods to create sub-widgets.

See also `dialog()`.

void **updateWidget** (int *i*, *sh::ui::FilePropertyDialogTabActionsView* \**widget*,  
*sh::filesystem::Operation* \**op*) **override**

Populates the widget for the *i*-th property with actual content. .

QString **titleWithoutMnemonic** ()

The tab title without mnemonic.

void **refresh** ()

Refreshes the complete content.

Typically called after some user actions in a property widget require that all the other content must be refreshed.

QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> **nodes** ()

The nodes to show.

*FilePropertyDialogTabActionsView* \***widgetAt** (int *i*)

Returns the widget for the *i*-th property.

int **widgetCount** ()

Returns the number of widgets.

std::shared\_ptr<*FilePropertyDialog*> **dialog** ()

Returns the associated dialog.

## Public Static Functions

void **doInitialize** ()

void **doShutdown** ()

**class FilePropertyDialogTabGeneral** : public *sh::ui::FilePropertyDialogTab*  
*#include <filepropertydialogtabgeneral.h>* Properties dialog tab for general infos.



## Public Functions

**FilePropertyDialogTabGeneral** ()

QString **title** () **override**

The tab title. .

QList<QString> **properties** () **override**

Returns the list of property labels (one entry for each widget).

Each property is displayed with a header and a widget (created in createWidget) with some content (populated in updateWidget).

void **populateWidget** (int *i*, *sh::ui::FilePropertyDialogTabActionsView* \**widget*) **override**

Populates the tab widget for the i-th property.

Typically uses the FilePropertyDialog::create\* methods to create sub-widgets.

See also *dialog*().

void **updateWidget** (int *i*, *sh::ui::FilePropertyDialogTabActionsView* \**widget*, *sh::filesystem::Operation* \**op*) **override**

Populates the widget for the i-th property with actual content. .

QString **titleWithoutMnemonic** ()

The tab title without mnemonic.

void **refresh** ()

Refreshes the complete content.

Typically called after some user actions in a property widget require that all the other content must be refreshed.

QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> **nodes** ()

The nodes to show.

FilePropertyDialogTabActionsView \***widgetAt** (int *i*)

Returns the widget for the i-th property.

int **widgetCount** ()

Returns the number of widgets.

std::shared\_ptr<FilePropertyDialog> **dialog** ()

Returns the associated dialog.

## Public Static Functions

void **doInitialize** ()

void **doShutdown** ()

**class FilePropertyDialogTabUnixPermissions** : public *sh::ui::FilePropertyDialogTab*  
*#include <filepropertydialogtabunixpermissions.h>* Properties dialog tab for unix permissions.

## Public Functions

**FilePropertyDialogTabUnixPermissions ()**

**QString title () override**

The tab title. .

**QList<QString> properties () override**

Returns the list of property labels (one entry for each widget).

Each property is displayed with a header and a widget (created in `createWidget`) with some content (populated in `updateWidget`).

**void populateWidget (int i, *sh::ui::FilePropertyDialogTabActionsView* \*widget) override**

Populates the tab widget for the i-th property.

Typically uses the `FilePropertyDialog::create*` methods to create sub-widgets.

See also *dialog()*.

**void updateWidget (int i, *sh::ui::FilePropertyDialogTabActionsView* \*widget, *sh::filesystem::Operation* \*op) override**

Populates the widget for the i-th property with actual content. .

**QString titleWithoutMnemonic ()**

The tab title without mnemonic.

**void refresh ()**

Refreshes the complete content.

Typically called after some user actions in a property widget require that all the other content must be refreshed.

**QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> nodes ()**

The nodes to show.

**FilePropertyDialogTabActionsView \*widgetAt (int i)**

Returns the widget for the i-th property.

**int widgetCount ()**

Returns the number of widgets.

**std::shared\_ptr<FilePropertyDialog> dialog ()**

Returns the associated dialog.

## Public Static Functions

**QString getUsername (int uid)**

**QString getGroupName (int gid)**

**QMap<int, QString> getAllUsers ()**

**QMap<int, QString> getAllGroups ()**

**void doInitialize ()**

**void doShutdown ()**

## Private Functions

```
void slot_buttontriggered (int i)
QString userPermissionsDescription (int flags, int rflag, int wflag, int xflag)
class ActionChange : public sh::actions::ActionActionItem
```

## Public Functions

```
ActionChange (std::shared_ptr<sh::filepropertydialogtabs::FilePropertyDialogTabUnixPermissions>
               tab)
void execute ()
    Executes this action.
void execute (sh::actions::ActionExecutionInfo *info)
    Executes this action.
QKeySequence shortcuthint ()
    Returns the keyboard shortcut for triggering this action.
bool shortcuthintTriggersOnCurrentDirectory ()
    Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the
    entry selection).
void setShortcuthint (QKeySequence shortcut, bool triggersOnCurrentDirectory =
                     false)
    Sets the keyboard shortcut for triggering this action.
QString text ()
    Returns the displayed text for this action.
QString icon ()
    Returns the icon for this action.
bool enabled ()
    Checks if this action is enabled.
bool isChecked ()
    Checks if this action is checked (has a cross in the ui).
bool isCheckable ()
    Checks if this action is checkable (can have a cross in the ui).
int defaultActionPrecedence () const
    Returns the precedence for becoming the default action of a parent submenu.
    This e.g. leads to a bold label.
    The action with the highest value becomes the default. Only values >0 will be considered. See
    ActionDefaultPrecedenceValues for reference values.
void setText (QString text)
    Sets the displayed text.
void setIcon (QString icon)
    Sets the icon.
void setEnabled (bool enabled)
    Sets if the item is enabled.
```

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<AbstractActionItem> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<AbstractActionItem> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

**class FilePropertyDialogTabWindows** : public *sh::ui::FilePropertyDialogTab*  
*#include <filepropertydialogtabwindows.h>* Properties dialog tab for unix permissions.

## Public Functions

QString **title** () = 0  
The tab title. .

QString **titleWithoutMnemonic** ()  
The tab title without mnemonic.

QList<QString> **properties** () = 0  
Returns the list of property labels (one entry for each widget).

Each property is displayed with a header and a widget (created in `createWidget`) with some content (populated in `updateWidget`).

void **populateWidget** (int *i*, FilePropertyDialogTabActionsView \**widget*) = 0  
Populates the tab widget for the *i*-th property.

Typically uses the `FilePropertyDialog::create*` methods to create sub-widgets.

See also *dialog()*.

void **updateWidget** (int *i*, FilePropertyDialogTabActionsView \**widget*,  
*sh::filesystem::Operation* \**op*) = 0  
Populates the widget for the *i*-th property with actual content. .

void **refresh** ()

Refreshes the complete content.

Typically called after some user actions in a property widget require that all the other content must be refreshed.

QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> **nodes** ()

The nodes to show.

FilePropertyDialogTabActionsView \***widgetAt** (int *i*)

Returns the widget for the *i*-th property.

int **widgetCount** ()

Returns the number of widgets.

std::shared\_ptr<FilePropertyDialog> **dialog** ()

Returns the associated dialog.

**namespace filesystem**

Filesystem model and auxiliary logic.

## Enums

**enum SizeFormatting**

*Values:*

**enumerator SizeFormattingModePrefixes**

**enumerator SizeFormattingModePlainBytes**

**enum FilesystemNodeType**

Enumeration of types a *sh::filesystem::FilesystemNode* can have.

*Values:*

**enumerator NONE** = 0

No type specified or does not exist.

**enumerator File** = 1

Usual file.

**enumerator FIRSTTYPE** = *File*

The first type (used for generating int ranges).

**enumerator Directory**

Directory.

**enumerator Link**

Link.

**enumerator Unknown**

Unknown.

**enumerator LASTPHYSICALTYPE** = *Unknown*

The last physical type, excluding internal magic types (used for generating int ranges).

**enumerator SpecialTreeOnlyDirectory**

Special kind of directory which behaves differently.

**enumerator Invalid**

This return type means that the type can't be determined. It should not occur in typical situations.

**enumerator LASTTYPE** = *SpecialTreeOnlyDirectory*

The last type (used for generating int ranges).

**class AdhocFilesystemNodeList** : public *sh::filesystem::FilesystemNodeList*  
#include <filesystemodelist.h> This *FilesystemNodeList* subclass is used for spontaneously fetching a fresh list of node children.

Node additions and removals will not touch the model.

Node instances inside it will be deleted together with this list.

## Public Functions

**AdhocFilesystemNodeList** ()

Is intended to be directly constructed from everywhere.

**~AdhocFilesystemNodeList** ()

void **addItem** (QSet<std::shared\_ptr<*FilesystemNode*>> *nodes*)

void **addItem** (std::shared\_ptr<*FilesystemNode*> *node*)

void **removeItems** (QSet<std::shared\_ptr<*FilesystemNode*>> *nodes*)

void **removeItem** (std::shared\_ptr<*FilesystemNode*> *node*)

void **resetItems** (*sh::filesystem::FilesystemNodeType* *type*, QSet<std::shared\_ptr<*FilesystemNode*>> *nodes*)

std::shared\_ptr<*FilesystemNode*> **mynode** ()

Returns the node which owns this children list. The result may be 0 for non model-backed lists.

void **addItem** (QSet<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> *nodes*) = 0

Adds items to this list. In a model-backed list, this triggers the internal model mounting calls. It is not allowed to call this method twice with the same node.

void **addItem** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> *node*) = 0

Adds an item to this list. In a model-backed list, this triggers the internal model mounting calls. It is not allowed to call this method twice with the same node.

void **removeItems** (QSet<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> *nodes*) = 0

Removes nodes from this list. In a model-backed list, this triggers the internal model unmounting calls. It is not allowed to call this method with a node, which is not contained in that list.

void **removeItem** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> *node*) = 0

Removes a node from this list. In a model-backed list, this triggers the internal model unmounting calls. It is not allowed to call this method with a node, which is not contained in that list.

void **resetItems** (*sh::filesystem::FilesystemNodeType* *type*, QSet<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> *nodes*) = 0

Resets the list for a given node type to a given new list of children nodes.

bool **contains** (std::shared\_ptr<*FilesystemNode*> *node*)

Checks if this list contains a given node.

const QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> \***nodes** ()

Returns the list of nodes currently stored in this instance.

**class DetailColumn** : public std::enable\_shared\_from\_this<*DetailColumn*>  
#include <detailcolumn.h> Abstract base class for a detail column (on filesystem nodes).

Those can e.g. be seen in the file list view, but can also be queried internally by other places in code.

It encapsulates the retrieval logic and metadata for one piece of additional information a *sh::filesystem::FilesystemNode* can have (e.g. the filesize). Retrieving the values is designed to be asynchronous. Each instance represents one column, while the actual logic is implemented in subclasses. For a new detail column, subclass this class and implement at least `determineValue`.

Subclassed by *sh::detailcolumns::DetailColumnCustomAttributes*, *sh::detailcolumns::DetailColumnDirectSymlinkTarget*, *sh::detailcolumns::DetailColumnExtendedAttributes*, *sh::detailcolumns::DetailColumnFilesize*, *sh::detailcolumns::DetailColumnMimetype*, *sh::detailcolumns::DetailColumnMtime*, *sh::detailcolumns::DetailColumnResolvedSymlink*, *sh::filesystemhandlers::SharcFilesystemHandler::ArchivedSizeDetailColumn*, *sh::scripting::api::ApiDetailColumn*

## Public Functions

QString **name** ()

The internal unique name.

QString **displayName** ()

The display name.

bool **isValueAvailable** (std::shared\_ptr<*FilesystemNode*> node)

Checks if a value for this detail is available for one given node.

QVariant **value** (std::shared\_ptr<*FilesystemNode*> node, bool ignoreAged = false)

Returns the value for this detail for one given node.

### Parameters

- ignoreAged: If no value should be returned which is older than the last request (not useful for nearly all cases).

QString **displayValue** (std::shared\_ptr<*FilesystemNode*> node, const *sh::filesystem::FilesystemModelFileviewProxy* \*viewmodel)

Returns the stringified value for this details for one given node considering the configuration of a view.

bool **isVisible** ()

If this detail shall be a visible column in the view.

QVariant **requestValue** (std::shared\_ptr<*FilesystemNode*> node, *sh::filesystem::Operation* \*op = 0, bool withNodeValues = false, bool withOperationsCache = true)

Requests to determine the value for this detail for one given node. Blocks until the value is available. Get it with the value method afterwards.

bool **sort\_doTypediff** ()

If sorting should separate files and directories.

int **sort\_order** (std::shared\_ptr<*FilesystemNode*> n1, std::shared\_ptr<*FilesystemNode*> n2)

The sorting order.

uint **displayIndex** ()

Controls which place this column should get in the user interface.

QVariant **computeValue** (std::shared\_ptr<*FilesystemNode*> node, *sh::filesystem::Operation* \*op)

Returns a freshly determined value for this column and the given node. It neither asks nor populates any caches or storages.

int **defaultWidth** ()

bool **isRightAligned** ()

**~DetailColumn()**

void **applyValueByEurl** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*,  
*sh::filesystem::Operation* \**op*, QVariant *value*)

Applies the detail value to a given eurl (without using any caches).

Used for transferring some details when a file transfer occurs.

Optionally implement the one of the applyValueBy... methods and register the instance with *sh::filesystem::FilesystemOperation::addTransferrableDetailColumn*. For performance reasons, you should decide to implement applyValueByEurl.

void **applyValueByNode** (std::shared\_ptr<*FilesystemNode*> *node*, *sh::filesystem::Operation*  
\**op*, QVariant *value*)

Applies the detail value to a given node (without using any caches).

Used for transferring some details when a file transfer occurs.

Optionally implement the one of the applyValueBy... methods and register the instance with *sh::filesystem::FilesystemOperation::addTransferrableDetailColumn*. For performance reasons, you should decide to implement applyValueByEurl.

## Public Static Functions

QVariant **VALUE\_UNAVAILABLE** ()

A special value expressing that the value is not yet available.

void **registerKnownDetailColumn** (QString *name*, std::shared\_ptr<*DetailColumn*> *column*)

std::shared\_ptr<*DetailColumn*> **findKnownDetailColumn** (QString *name*)

## Public Static Attributes

const uint **DISPLAYINDEX\_CORE** = 10000

const uint **DISPLAYINDEX\_INTERESTING** = 20000

const uint **DISPLAYINDEX\_EXOTIC** = 30000

## Private Members

QString **\_displayName**

QString **\_name**

uint **\_displayIndex**

bool **\_sort\_doTypediff**

int **\_defaultWidth**

bool **\_isRightAligned**



## Private Static Attributes

QMap<QString, std::shared\_ptr<*DetailColumn*>> **\_knownDetailColumns**

QMutex **\_knownDetailColumnsMutex**

## Friends

**friend class** sh::filesystem::FileSystemNode

**class Eurl** : public std::enable\_shared\_from\_this<*Eurl*>

*#include <eurl.h>* Filesystem paths.

A Extended Uniform Resource Locator is something like a [URL](#).

It is more general since it can also be nested. It is something like `zip:/{}/zippedfolder/zippedfile`.

You get *Eurl* instances only from the factory methods. There will never be two instances with the same textual value.

Please note: Instances can be associated with any kind of elements in the filesystem (files, directories, links, ...). It might also point to something which does not exist at all. The documentation sometimes explicitly makes a difference between those kinds (files, directories, links, ...; often called ‘node type’). But it often uses the term ‘file’ implicitly while meaning all kinds of elements; assuming that e.g. a directory is just a special kind of a file. It should be clear from the particular context which meaning applies.

## Public Functions

QString **asString()** **const**

Returns the textual value. This is what *Eurl.fromString* would expect as parameter.

QString **hostname()** **const**

Returns the hostname part (from the outer url of this *Eurl*). Examples: "livingroom-pc" for `smb://livingroom-pc/foo/bar/baz`. "" for `.`

QString **path()** **const**

Returns the path part (from the outer url of this *Eurl*). Examples: `"/foo/bar/baz"` for `smb://livingroom-pc/foo/bar/baz`. `"/foo/bar/baz"` for `.` `"/"` for `.`

QString **basename()** **const**

Returns the last path segment. This is the text behind the last slash. Examples: "baz" for `.` "" for `.`

QString **scheme()** **const**

Returns the scheme (from the outer url of this shallot.Eurl). This is what comes before the `://`. Example: "file" for `.`

std::shared\_ptr<const *Eurl*> **outerUrl()** **const**

Returns a new *Eurl* containing only the outer part of this one (strips the embeddings). Example: `foobar://host/foo/bar/baz` for `foobar:[zip:/{}/d/e]//foo/bar/baz`.

std::shared\_ptr<const *Eurl*> **outermostInnerEurl()** **const**

Returns a new *Eurl* containing only the embedding of this one. Example: `zip:/{}/d/e` for `foobar:[zip:/{}/d/e]//foo/bar/baz`.

`std::shared_ptr<const Eurl> withAppendedSegment (QString basename) const`  
Returns a new *Eurl* from this one with “/basename” appended. The parameter is assumed to be a single path segment.

`std::shared_ptr<const Eurl> withAppendedSegments (QString path) const`  
Returns a new *Eurl* with path segments “/pa/t/h/” appended. The parameter may contain “/”s for dividing path segments.

`std::shared_ptr<const Eurl> root () const`  
Returns the root *Eurl* from this one. Example: `zip:>[]//` for `zip:>[]//foo/bar/baz`.

`std::shared_ptr<const Eurl> enwrapWithOuterUrl (QString scheme, QString hostname,  
QString path) const`  
Returns a new *Eurl* containing this one packed as embedding and new outer parts *scheme*, *hostname* and *path*. Example: `scheme:>[]/hostname/p/a/t/h` for `.`

`std::shared_ptr<const Eurl> parentSegment () const`  
Returns the parent *Eurl*. At first, this traverses path segments. For a root path *eurl* with embeddings, it returns the embedding. If none are available, it returns 0. Examples: `foo://host/foo` for `foo://host/foo/bar`. `foo://host/` for `foo://host/boo`. `foo:[bar://host/goo]/anotherhost/` for `foo:[bar://host/goo]/anotherhost/boo`. `bar://host/foo` for `foo:[bar://host/foo]/host/`. 0 for `foo://host/`.

`bool hasInnerUrls () const`  
Checks if this *Eurl* has embeddings. Examples: `true` for `foo:[bar:///foo]/host/`. `false` for `foo://host/`.

`bool outerUrlsIsRootDirectory () const`  
Checks if this *Eurl* is a root path (with or without embeddings). Examples: `true` for `foo://host/`. `false` for `foo://host/a`. `true` for `foo:[bar:///goo]/host/`. `false` for `foo:[bar:///goo]/host/a`. `true` for `foo:[bar:///goo]//`. `false` for `foo:[bar:///goo]//a`.

`bool hasParentSegment () const`  
Checks if this *Eurl* has a parent segment. This indicates if *Eurl.parentSegment* would return 0.

`bool isPrefixOf (std::shared_ptr<const Eurl> longer) const`  
Checks if this *Eurl* is a prefix of another one. This is not an equivalent to a string comparison but it checks parent relationships according to *Eurl.parentSegment*.

**Eurl** (QString *eurlstring*)  
Constructed only by the infrastructure and made available otherwise.

**~Eurl** ()

## Public Static Functions

`std::shared_ptr<const Eurl> fromString (QString eurlstring)`  
Constructs a new *Eurl* by string (what *Eurl.asString* would return).

`std::shared_ptr<const Eurl> create (QString scheme, QString hostname, QString path)`  
Constructs a new *Eurl* by scheme name, hostname and a path.

`std::shared_ptr<const Eurl> create (QString scheme, const Eurl *inner, QString hostname, QString path)`  
Constructs a new *Eurl* by scheme name, an inner *eurl*, a hostname and a path.

`void filenameCheck (QString filename)`  
Checks if a name is a valid filename. If not, *EurlMisformattedException* is thrown.

```
void doInitialize ()
```

```
void doShutdown ()
```

### Public Static Attributes

```
const QChar WRAPPER_BEGIN = '['
```

The character marking the begin of an embedding.

```
const QChar WRAPPER_END = ']'
```

The character marking the end of an embedding.

```
const QString FORBIDDEN_FILENAME_CHARACTERS = QString("/")
```

Characters which are forbidden in filenames.

### Private Members

```
const QString _eurlstring
```

```
std::shared_ptr<const Eurl> _cache_parentsegment = 0
```

```
std::shared_ptr<const Eurl> _cache_outerurl = 0
```

```
bool _cache_outerurl_isthis = false
```

```
std::shared_ptr<const Eurl> _cache_outermostinnereurl = 0
```

```
std::shared_ptr<const Eurl> _cache_root = 0
```

```
bool _cache_root_isthis = false
```

```
QString _cache_basename
```

```
QString _cache_hostname
```

```
QString _cache_path
```

```
QString _cache_scheme
```

### Private Static Functions

```
QString check_scheme (QString scheme)
```

```
QString check_inner (QString inner)
```

```
QString check_hostname (QString hostname)
```

```
QString check_path (QString path)
```

```
QString check_filename (QString filename)
```

```
QString _escape (QString s)
```

```
QString _unescape (QString s)
```

```
std::shared_ptr<const Eurl> createNOCHECK (QString scheme, const Eurl *inner,
                                           QString hostname, QString path)
```

```
std::shared_ptr<const Eurl> fromStringNOCHECK (QString eurlstring)
```

### Private Static Attributes

```
QHash<QChar, QString> _escapemap
QMutex _escapemapmutex
QMutex _mutex
QHash<QString, std::weak_ptr<const Eurl>> _eurluniverse
```

```
class EurlMisformattedException: public sh::exceptions::ArgumentException
    #include <eurl.h> Shallot exception for invalid input in Eurl creation methods.
```

### Public Functions

```
EurlMisformattedException (sh::exceptions::ExceptionData data)
QString message () const
QString name () const
QString classes () const
QString details () const
QString callstack () const
QString auxiliary () const
QString customValue (QString key) const
bool isRuntimeException () const
bool isProgramException () const
bool isRetryable () const
bool isResumeable () const
bool isDetailsAreInteresting () const
int autoRetryRecommendedIn () const
void setResumeable (bool v)
void setReTryable (bool v)
void setCustomValue (QString key, QString value)
bool isClass (QString classname)
sh::exceptions::ExceptionData data ()
```

### Public Static Functions

```
template<class Handler>
std::shared_ptr<RegisterHandler<Handler>> createRegisterHandler (Handler    han-
                                                                dler,      QS-
                                                                tack<Handler>
                                                                *stack)

void executeGuarded (std::function<void>
    > fctint flags = 0Executes some code with some standard exection handling around.
```

#### Parameters

- `fcn`: The code to execute guarded.
- `flags`: Flags of *ExecuteGuardFlag* for choosing the behavior.

```
void executeGuarded_errorpanel (std::function<void>
    > fcn, int flags = 0)
```

```
QString _demangle (QString l)
```

```
void exceptionDialog (sh::exceptions::Exception &e, bool *doRetry)
```

## Public Static Attributes

```
const QString UNDEFINED_ERROR_OCCURRED = QObject::tr("An unspecified error occurred.")
```

```
const QString SHALLOT_MUST_CLOSE_TEXT = QObject::tr("Your Shallot process is disturbed by an error and ne
```

```
const QString Value_isShallotException = "_isShallotException"
```

```
class FilesystemHandler : public QObject
```

#include <filesystemhandler.h> Abstract base class for a custom filesystem handler.

Subclass and register it for implementing a new virtual filesystem, which lets new nodes appear somewhere in the filesystem tree and controls how to handle them.

Use *sh::filesystem::FilesystemHandlerRegister* for registration.

For executing some actions or checks on a filesystem, you should not use those handlers directly, but the higher-level *sh::filesystem::FilesystemOperation* class.

Subclassed by *sh::filesystemhandlers::ArchiveFilesystemHandler*,  
*sh::filesystemhandlers::GnomeIOFilesystemHandler*, *sh::filesystemhandlers::LocalFilesystemHandler*,  
*sh::filesystemhandlers::SharcFilesystemHandler*, *sh::scripting::api::ApiFilesystemHandler*,  
*sh::search::SearchFilesystemHandler*

## Public Functions

```
FilesystemHandler (sh::filesystem::FilesystemModel *model)
```

Is (for subclasses) intended to be directly constructed and registered once.

```
~FilesystemHandler ()
```

```
void itemlist (sh::filesystem::Operation *op, std::shared_ptr<const
    sh::filesystem::Eurl> eurl, sh::filesystem::FilesystemNodeType type,
    sh::filesystem::FilesystemNodeListEditor *list) = 0
```

Determine a list of subelements in a certain directory.

```
void configureItems (sh::filesystem::Operation *op, QList<std::shared_ptr<FilesystemNode>>
    nodes)
```

Configure newly created nodes (e.g. setting another icon or changing the display name).

```
sh::filesystem::FilesystemNodeType getType (sh::filesystem::Operation *op,
    std::shared_ptr<const sh::filesystem::Eurl>
    eurl) = 0
```

Determine if a file is regular, or a dir, or a link, ...

```
qint64 getSize (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
    eurl) = 0
```

Determine the size of a file.

```
QDateTime getMtime (sh::filesystem::Operation *op, std::shared_ptr<const
    sh::filesystem::Eurl> eurl) = 0
```

Determine the mtime of a file.

```

void setMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
               eurl, QDateTime time) = 0
    Set the mtime of a file.

QString getMimeType (sh::filesystem::Operation *op, std::shared_ptr<const
                   sh::filesystem::Eurl> eurl) = 0
    Determine mime type of a file.

QString getLinkTarget (sh::filesystem::Operation *op, std::shared_ptr<const
                      sh::filesystem::Eurl> eurl) = 0
    Resolve a link.

QList<QString> listExtendedAttributes (sh::filesystem::Operation *op,
                                       std::shared_ptr<const sh::filesystem::Eurl>
                                       eurl)
    Fetches the list of available extended attributes for an entry.

quint64 getExtendedAttributeSize (sh::filesystem::Operation *op,
                                   std::shared_ptr<const sh::filesystem::Eurl>
                                   eurl, QString attribute)
    Returns the size of value for one extended attribute for an entry.

QByteArray getExtendedAttribute (sh::filesystem::Operation *op,
                                  std::shared_ptr<const sh::filesystem::Eurl>
                                  eurl, QString attribute)
    Returns the value for one extended attribute for an entry.

void setExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
                           sh::filesystem::Eurl> eurl, QString attribute, QByteArray
                           value)
    Sets the value for one extended attribute for an entry.

void removeExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
                              sh::filesystem::Eurl> eurl, QString attribute)
    Removes one extended attributes for an entry.

QMap<QString, QString> getCustomAttributes (sh::filesystem::Operation
                                              *op, std::shared_ptr<const
                                              sh::filesystem::Eurl> eurl)
    Returns the custom attributes for an entry.

    In contrast to extended attributes, the custom attributes are not directly stored in the filesystem
    this way, but handle implementation specific aspects (like permissions).

void setCustomAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
                          sh::filesystem::Eurl> eurl, QString attribute, QString value)
    Sets the value for one custom attribute for an entry.

    See also getCustomAttributes.

bool canGetFileContent (sh::filesystem::Operation *op, std::shared_ptr<const
                        sh::filesystem::Eurl> eurl) = 0
    Returns if it is allowed to get the content of a certain file.

std::shared_ptr<QIODevice> getFileContent (sh::filesystem::Operation
                                           *op, std::shared_ptr<const
                                           sh::filesystem::Eurl> eurl) = 0
    Get the content of a file.

bool canCreateDirectory (sh::filesystem::Operation *op, std::shared_ptr<const
                          sh::filesystem::Eurl> eurl) = 0
    Returns if it is allowed to create subdirectories in a certain directory.

```

```

void createDirectory (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) = 0
    Create a directory.

bool canCreateLink (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) = 0
    Returns if it is allowed to create a link in a certain directory.

void createLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                eurl, QString target) = 0
    Create a link.

bool canCreateFile (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) = 0
    Returns if it is allowed to create files in a certain directory.

void createFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                eurl, QIODevice *content, HandlerTransfer *handlertransfer) = 0
    Creates a file with some content.

    It may use handlertransfer for some better integration, like cancel support and progress
    monitoring.

bool canDeleteItem (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) = 0
    Returns if it is allowed to delete a certain file/directory/link/...

void deleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                eurl) = 0
    Delete a file/directory/link/...

void deleteDirectoryIfEmpty (sh::filesystem::Operation *op, std::shared_ptr<const
                            sh::filesystem::Eurl> eurl)
    Deletes a directory only if it is empty.

bool canRenameItem (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> src) = 0
    Returns if it is allowed to rename a certain file, directory, link, ... (see renameItem).

void renameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                src, QString destpath) = 0
    Renames an item to another path.

    It can be a simple filename change or also changes in the deeper path segments.

QList<std::shared_ptr<sh::actions::AbstractActionItem>> getActions (const
                                                                QList<std::shared_ptr<const
                                                                sh::filesystem::Eurl>>
                                                                eurls) = 0
    Returns a list of actions (see former example) for showing for certain files.

bool requiresResolvedLinks ()
    Returns if this handler requires to be used by the engine with resolved links.

void viewEnteredDirectory (std::shared_ptr<const sh::filesystem::Eurl>)
    Callback for preparing stuff when the view entered some directory (e.g. for enabling watchers).
    See also viewLeftDirectory.

void viewLeftDirectory (std::shared_ptr<const sh::filesystem::Eurl>)
    Callback for preparing stuff when the view left some directory. See also viewEnteredDirectory.

bool isWellKnownScheme ()
    Returns if the scheme for this handler is a well known one (which external programs typically
    would understand).

```

```
void search (sh::filesystem::Operation *op, std::shared_ptr<const  
    sh::filesystem::Eurl> eurl, std::function<void> std::shared_ptr<const  
    sh::filesystem::Eurl>, QList<std::shared_ptr<sh::search::SearchCriterion>>,  
    sh::filesystem::FilesystemNodeType ftype  
> addfile, std::function<void>std::shared_ptr<const sh::filesystem::Eurl>> visitdir,  
    QList<std::shared_ptr<sh::search::SearchCriterion>> criteria)Helps searching for files.
```

Override this method with a behavior identical to the default, if a specialized implementation can be much faster than the default one.

```
void customizeUi (std::shared_ptr<sh::filesystem::FilesystemNode> node, bool *showSearch-  
    Panel)
```

Creates a custom gui, which becomes part of the fileview.

```
sh::filesystem::FilesystemModel *model ()
```

A convenience shortcut to the *sh::filesystem::FilesystemModel* (a singleton).

```
class HandlerTransfer
```

```
#include <filesystemhandler.h> Subclassed by sh::filesystem::FilesystemOperation::MyHandlerTransfer,  
sh::filesystem::FilesystemOperationTransfers::SingleStepMonitor
```

### Public Functions

```
void respectCancel ()
```

```
void incrementTransferredBytes (qint64 donebytes)
```

```
~HandlerTransfer ()
```

```
class FilesystemHandlerRegister : public QObject, public sh::base::Singleton
```

```
#include <filesystemhandlerregister.h> A register of filesystem handlers.
```

Each active (i.e. referred to by existing nodes) instance of *sh::filesystem::FilesystemHandler* must be registered here.

### Public Functions

```
void addHandler (QString scheme, std::shared_ptr<sh::filesystem::FilesystemHandler> han-  
    dler)  
Registers a filesystem handler.
```

#### Parameters

- *scheme*: The scheme (very first part of a *sh::filesystem::Eurl*) for which the handler is responsible for.
- *handler*: The filesystem handler.

```
std::shared_ptr<sh::filesystem::FilesystemHandler> findHandler (QString scheme)  
Finds a filesystem handler by scheme.
```

```
void doInitialize ()  
Executes singleton initialization.
```

```
void doShutdown ()  
Executes singleton shutdown.
```

```
void shutdown ()  
Shutdown down this singleton.
```

```
bool isAlive ()  
Returns if this singleton is alive (true until its shutdown begins).
```



## Private Functions

**FilesystemHandlerRegister** ()

## Private Members

QHash<QString, std::shared\_ptr<*sh::filesystem::FilesystemHandler*>> **handlers**

QMutex **mutex**

**class FilesystemModel** : public QAbstractItemModel, public *sh::base::Singleton*  
*#include <filesystemmodel.h>* The filesystem model.

This is the engine which creates and manages *sh::filesystem::FilesystemNode* nodes. Filesystem nodes are used on many places for all kinds of operations.

## Public Functions

std::shared\_ptr<*sh::filesystem::FilesystemNode*> **rootNode** ()

The *sh::filesystem::FilesystemNode* which is the root node of the entire model. It is the parent for all toplevel nodes.

QVariant **data** (const QModelIndex &*index*, int *role* = Qt::DisplayRole) const

Qt::ItemFlags **flags** (const QModelIndex &*index*) const

QVariant **headerData** (int *section*, Qt::Orientation *orientation*, int *role* = Qt::DisplayRole) const

QModelIndex **index** (int *row*, int *column*, const QModelIndex &*parent* = QModelIndex()) const

QModelIndex **parent** (const QModelIndex &*index*) const

int **rowCount** (const QModelIndex &*parent* = QModelIndex()) const

int **columnCount** (const QModelIndex &*parent* = QModelIndex()) const

*sh::filesystem::FilesystemModelFileviewProxy* \***createFileviewProxy** (QModelIndex

*root*, bool

*withtooltip*,

std::function<void> std::shared\_ptr<*sh::fi*

> *onBecomesInvalid* Creates a *sh::filesystem::FilesystemModelFileviewProxy* for presenting the content of a directory.

QList<QPersistentModelIndex> **findIndexesForEurl** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*)

Get a list of qt model indexes for a *sh::filesystem::Eurl*.

If nodes for this eurl are unknown to the model so far, it tries to build them. In typical cases, this list either contains one element, or is empty if the filesystem handlers decide that this file does not exist. But in some cases, there is also more than one index for one *sh::filesystem::Eurl* (when the *sh::filesystem::Eurl* appears on more than one place in the tree).

QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> **findNodesForEurl** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*)

Get a list of *sh::filesystem::FilesystemNode* for a *sh::filesystem::Eurl*.

If it is unknown to the model so far, it tries to build them. In typical cases, this list either contains one element, or is empty if the filesystem handlers decide that this file does not exist. But in some

cases, there is also more than one node for one *sh::filesystem::Eurl* (when the *sh::filesystem::Eurl* appears on more than one place in the tree). It only returns nodes, which are ‘alive’, i.e. which have a living parent and which are a child of this parent.

`QList<std::shared_ptr<sh::filesystem::FileSystemNode>> tryGetNodesForEurl (std::shared_ptr<const sh::filesystem::Eurl> eurl)`

Returns a list of *sh::filesystem::FileSystemNode* for a *sh::filesystem::Eurl*. It only considers the current state of the in-memory model. It will only return nodes which are already known to the model so far. This operation is cheaper and handling only the known nodes is sufficient in many situations. In typical cases, this list either contains one element, or is empty. But in some cases, there is also more than one node for one *sh::filesystem::Eurl* (when the *sh::filesystem::Eurl* appears on more than one place in the tree). It only returns nodes, which are ‘alive’, i.e. which have a living parent and which are a child of this parent.

`std::shared_ptr<sh::filesystem::FileSystemNode> getNodeForIndex (const QModelIndex index) const`

Returns the *sh::filesystem::FileSystemNode* for a qt model index (in the main model).

`QModelIndex getIndexForNode (std::shared_ptr<sh::filesystem::FileSystemNode> node)`

Returns a qt model index (in the main model) for a *sh::filesystem::FileSystemNode*.

`std::shared_ptr<sh::filesystem::FileSystemNode> createFileSystemNode (std::shared_ptr<const sh::filesystem::Eurl> eurl, sh::filesystem::FileSystemHandler *handler, sh::filesystem::FileSystemNodeType nodetype, bool isHidden, std::shared_ptr<sh::filesystem::FilesystemNode> parent, bool doinsert = true, bool showInitialLoadingLabel = true)`

Creates a new *sh::filesystem::FileSystemNode* for placing it into the model.

If such a node (with the same eurl for the same parent node) does not exist, it generates a new one. If there already is such a node alive, but currently not placed in the model, it recycles this one. This can happen when references exist to a node which is not yet inserted or which is removed meanwhile. It is not allowed to call this method when such a node already exists in the model.

Use this function for getting a *sh::filesystem::FileSystemNode*, which is to be added to the model now or later. Depending on some parameter values, a call directly adds the node to the filesystem model (not e.g. when `doinsert` is `false` or `parentnode` is 0) It is typically used within a *sh::filesystem::FileSystemHandler* implementation.

```
std::shared_ptr<sh::filesystem::FileSystemNode> getOrCreateFileSystemNode (std::shared_ptr<const
sh::filesystem::Eurl>
neweurl,
sh::filesystem::FileSystemHandl
*han-
dler,
sh::filesystem::FileSystemNode?
node-
type,
bool
isHid-
den,
std::shared_ptr<sh::filesystem::
parent,
bool
doinsert
= true,
bool
showIni-
tial-
Load-
ingLabel
= true,
bool
*pIsNew
= 0)
```

Returns a *sh::filesystem::FileSystemNode* for using it as a child node in the model.

It either creates a new one, if there currently is no node for this eurl in this parentnode, or returns the existing one. Even for existing ones, this call can change the nodetype of that node.

Depending on some parameter values, a call directly adds the node to the filesystem model (not e.g. when doinsert is *false* or parent is 0).

```
void refreshData (std::shared_ptr<const sh::filesystem::Eurl> eurl, bool forceFindParent =
false, bool withDetails = true)
```

Request to refresh the internal model information for a *sh::filesystem::Eurl*. This may be a place which is already known (then a change of some metadata or the removal is detected) or a formerly unknown place (then new nodes get inserted in the model).

```
void addOpenNodeHelper (int index, std::function<QList<std::shared_ptr<sh::filesystem::FileSystemNode>>>() std:::sh::filesystem::Eurl>
```

> *openNodeHelper* Registers a helper method for ‘opening’ (mounting, activating, ...) locations on demand.

Only used in very rare cases.

```
~FileSystemModel ()
```

```
void doInitialize ()
```

Executes singleton initialization.

```
void doShutdown ()
```

Executes singleton shutdown.

```
void shutdown ()
```

Shutdown down this singleton.

```
bool isAlive ()
```

Returns if this singleton is alive (true until its shutdown begins).

## Private Functions

**FilesystemModel** ()

**QList**<std::shared\_ptr<sh::filesystem::FilesystemNode>> **\_findNodesForEurl\_helper** (std::shared\_ptr<const sh::filesystem::Eurl> eurl)

**QList**<std::shared\_ptr<sh::filesystem::FilesystemNode>> **openRootEurl** (std::shared\_ptr<const sh::filesystem::Eurl> eurl)

void **\_subitemFetchingStateChanged** (std::shared\_ptr<sh::filesystem::FilesystemNode> node, bool value)

## Private Members

**QList**< std::function< QList< std::shared\_ptr< sh::filesystem::FilesystemNode > >

**QMap**< int, std::function< QList< std::shared\_ptr< sh::filesystem::FilesystemNode

std::shared\_ptr<sh::filesystem::FilesystemNode> **rootnode**

**QMutex** **mutex**

**QHash**<std::shared\_ptr<const sh::filesystem::Eurl>, std::weak\_ptr<sh::filesystem::FilesystemNode>> **eurl2node**

**QMutex** **eurl2nodemutex**

**QMutex** **\_nodeDataMutex**

**QMutex** **\_openNodeHelpersMutex**

## Friends

**friend class** FilesystemNode

**friend class** LoadOnDemandPlaceholderFilesystemNode

**friend class** ModelBackedFilesystemNodeList

**friend class** ::sh::filesystem::DetailColumn

**class FilesystemModelDirectoryTreeProxy** : public QSortFilterProxyModel, public sh::base::Singleton  
#include <filesystemmodeldirectorytreeproxy.h> A filesystem proxy model for the directory tree.

It has a special sorting and filtering behavior.

Used internally, mostly for the user interface.

## Public Functions

void **enforceVisibility** (std::shared\_ptr<FilesystemNode> node)  
Marks a node as visible in the directory tree, even if it is a hidden node.

void **deEnforceVisibility** (std::shared\_ptr<FilesystemNode> node)  
Unmarks a node for being visible even if hidden (reverses *enforceVisibility*()).

void **doShutdown** ()  
Executes singleton shutdown.

```
void shutdown ()
    Shutdown down this singleton.

bool isAlive ()
    Returns if this singleton is alive (true until its shutdown begins).
```

### Private Functions

```
FilesystemModelDirectoryTreeProxy ()
```

```
class FilesystemModelDirectoryTreeProxyVisibilityEnforcements : public QObject, public s
    #include <filesystemmodeldirectorytreeproxyvisibilityenforcements.h> Maintains a list of currently
    visible directories (i.e. the current one in each view) and controls enforced visibility of them in the
    directory tree.
```

### Public Functions

```
void nodeEnteredView (std::shared_ptr<FilesystemNode> node)
    Called when a view enters the given directory node.

void nodeLeftView (std::shared_ptr<FilesystemNode> node)
    Called when a view leaves the given directory node.

void nodeCollapsedInTree (std::shared_ptr<FilesystemNode> node)
    Called when a directory node is collapsed in the directory tree.

void doShutdown ()
    Executes singleton shutdown.

void shutdown ()
    Shutdown down this singleton.

bool isAlive ()
    Returns if this singleton is alive (true until its shutdown begins).
```

### Private Functions

```
FilesystemModelDirectoryTreeProxyVisibilityEnforcements ()

bool _addenforcement (std::shared_ptr<FilesystemNode> node,
                      std::shared_ptr<FilesystemNode> hnode)
```

### Private Members

```
QMultiMap<std::shared_ptr<FilesystemNode>, QObject*> _enterednodesrepresentatives

QList<std::tuple<std::weak_ptr<FilesystemNode>, QList<std::shared_ptr<FilesystemNode>>>> hiddenForcedVis

class FilesystemModelFileviewProxy : public QSortFilterProxyModel
    #include <filesystemmodelfileviewproxy.h> A filesystem proxy model for a fileview.

    It regards the sorting and filtering behavior set up for the connected fileview.

    Used internally, mostly for the user interface.
```

## Public Functions

**FilesystemModelFileviewProxy** (QModelIndex *root*, bool *withtooltip*, QObject *\*parent* = 0)

Constructed only indirectly.

void **setSizeFormattingMode** (*SizeFormatting mode*)

*SizeFormatting* **getSizeFormattingMode** () const

void **setHiddenFilesVisible** (bool *v*)

bool **getHiddenFilesVisible** () const

QVariant **data** (const QModelIndex &*index*, int *role* = Qt::DisplayRole) const

void **triggerReloadData** ()

void **setThumbnail** (bool *enabled*, int *size* = 32)

bool **getThumbnailEnabled** ()

int **getThumbnailSize** ()

**class FilesystemModelSubtreeProxy** : public QAbstractItemModel

#include <filesystemmodelsubtreeproxy.h> A filesystem proxy model for setting a new root node to an existing fileview proxy.

Used internally, mostly for the user interface.

## Public Functions

**FilesystemModelSubtreeProxy** (QModelIndex *root*, *sh::filesystem::FilesystemModelFileviewProxy \*upperproxy*)

Constructed only indirectly.

QVariant **data** (const QModelIndex &*index*, int *role*) const

Qt::ItemFlags **flags** (const QModelIndex &*index*) const

QVariant **headerData** (int *section*, Qt::Orientation *orientation*, int *role* = Qt::DisplayRole) const

QModelIndex **index** (int *row*, int *column*, const QModelIndex &*parent* = QModelIndex()) const

QModelIndex **parent** (const QModelIndex &*index*) const

int **rowCount** (const QModelIndex &*parent* = QModelIndex()) const

int **columnCount** (const QModelIndex &*parent* = QModelIndex()) const

QModelIndex **mapFromSource** (const QModelIndex *mi*) const

QModelIndex **mapToSource** (const QModelIndex *mi*) const

## Signals

void **becomesInvalid** (std::shared\_ptr<sh::filesystem::FilesystemNode> *bestValid*)

## Private Members

bool **\_inChangeTransaction**

std::shared\_ptr<sh::filesystem::FilesystemNode> **\_rootnode**

QPersistentModelIndex **\_rootindex**

sh::filesystem::FilesystemModel \***mainmodel**

sh::filesystem::FilesystemModelFileviewProxy \***\_upperproxy**

**class FilesystemNode** : public QObject, public std::enable\_shared\_from\_this<FilesystemNode>  
*#include <filesystemnode.h>* A representation of a location in the filesystem tree.

It represents filesystem nodes like a file or a directory in *sh::filesystem::FilesystemModel*.

Please note: Instances can be associated with any kind of elements in the filesystem (files, directories, links, ...). It might also point to something which does not exist at all (see parentnode). The documentation sometimes explicitly makes a difference between those kinds (files, directories, links, ...; often called ‘node type’). But it often uses the term ‘file’ implicitly while meaning all kinds of elements; assuming that e.g. a directory is just a special kind of a file. It should be clear from the particular context which meaning applies.

Subclassed by *sh::filesystem::LoadOnDemandPlaceholderFilesystemNode*

## Public Functions

**~FilesystemNode** ()

std::shared\_ptr<const sh::filesystem::Eurl> **eurl** ()

Returns the *sh::filesystem::Eurl* entry address.

sh::filesystem::FilesystemModel \***model** ()

Convenience shortcut to the *sh::filesystem::FilesystemModel*.

QString **displayName** ()

Returns the display name (what the user interface displays).

void **setDisplayName** (QString *displayname*)

Sets the display name (what the user interface displays).

sh::filesystem::FilesystemHandler \***handler** ()

Returns the handler which is responsible for this node. This should be the same as *sh::filesystem::FilesystemHandlerRegister.findHandler(eurl.scheme)*.

int **nodetype** ()

Returns the FilesystemNodeType node type.

int **childnodeCount** ()

Returns the number of children nodes.

std::shared\_ptr<sh::filesystem::FilesystemNode> **childnode** (int *i*)

Returns i-th child node.

*sh::filesystem::ModelBackedFilesystemNodeList* \***childNodes** ()

Returns the list of children.

std::shared\_ptr<*sh::filesystem::FilesystemNode*> **parentnode** () **const**

Returns the parent node.

This is 0 if the node does not exist in the model (i.e. not yet inserted or removed afterwards). In most cases, this can be interpreted as ‘file currently does not exist’.

int **index** ()

Returns the index of this node in the parent’s list of children.

void **addChild** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> *node*)

Adds a new child. Not allowed to call more than once for a node.

void **removeChild** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> *node*)

Removes a new child. Not allowed to call more than once for a node.

bool **isFetchingSubitems** ()

Checks if currently subitems are fetched (i.e. the ‘loading’ node is shown).

QIcon **icon** ()

Returns the node icon.

void **setIcon** (QIcon *icon*)

Sets the node icon.

bool **isHidden** ()

Returns if the node is hidden.

Note: It’s not difficult for the user to also show the hidden items.

void **setHidden** (bool *v*)

Sets if the node is hidden.

See also *isHidden()*.

void **addDetail** (std::shared\_ptr<*sh::filesystem::DetailColumn*> *column*)

Adds a detail to this node.

std::shared\_ptr<*sh::filesystem::DetailColumn*> **getDetailColumnByIndex** (int *index*)

Returns the i-th detail column registered to this node.

int **getDetailColumnsCount** () **const**

Returns the number of detail columns registered to this node.

bool **isRootNode** () **const**

Checks if this is the root node of the model.

bool **isConfigured** () **const**

Checks if this node was already configured by a handler.

void **requestDetails** (bool *force* = true)

Requests to fetch details for this node.

bool **isAlive** () **const**

Checks if this node currently exists in the model.

QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> **linkDestinationNodes** ()

**const**  
Returns the list of link source nodes. If this node is a link, the link destination nodes may influence the behavior and appearance of this node.

This information does not come from inside but must be set from outside the model implementation.



void **setLinkDestinationNodes** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>>  
*linkdestinations*)

Sets the list of link destination nodes. If this node is a link, the link destination nodes may influence the behavior and appearance of this node.

void **requestContainedItems** (*sh::filesystem::FilesystemNodeType* type = *FilesystemNodeType::NONE*)

Requests to fetch the children of this node with help of the filesystem handler.

**FilesystemNode** (*sh::filesystem::FilesystemModel* \*model, std::shared\_ptr<const  
*sh::filesystem::Eurl*> eurl, QString displayname,  
*sh::filesystem::FilesystemHandler* \*handler, *FilesystemNodeType* node-  
type, bool ishidden)

Constructed only by the infrastructure and made available otherwise.

## Signals

void **removed** ()

Is emitted when this node is not placed in the tree anymore. This does not mean the physical deletion of the instance, but just means it is not alive from now on.

void **\_detailsAvailable** ()

Is emitted when values for detail columns arrived.

void **iconChanged** ()

Is emitted when the icon changed.

void **isHiddenChanged** ()

Is emitted when the hidden flag changed.

## Private Functions

void **setDetail** (std::shared\_ptr<*sh::filesystem::DetailColumn*> column, QVariant value)

int **getInsertPositionForDetailColumn** (std::shared\_ptr<*sh::filesystem::DetailColumn*>  
newCol)

## Friends

**friend class** *sh::filesystem::FilesystemModel*

**friend class** *sh::filesystem::DetailColumn*

**friend class** *LoadOnDemandPlaceholderFilesystemNode*

**friend class** *sh::filesystem::FilesystemModelFileviewProxy*

**friend class** *sh::filesystem::FilesystemModelSubtreeProxy*

**friend class** *sh::filesystem::ModelBackedFilesystemNodeList*

**class FilesystemNodeList** : public QObject

#include <*filesystemnodelist.h*> A data structure for children lists of filesystem nodes.

Different subclasses exist with different behaviors and use cases.

Subclassed by *sh::filesystem::AdhocFilesystemNodeList*, *sh::filesystem::ModelBackedFilesystemNodeList*

## Public Functions

void **addItem** (QSet<std::shared\_ptr<sh::filesystem::FilesystemNode>> nodes) = 0

Adds items to this list. In a model-backed list, this triggers the internal model mounting calls. It is not allowed to call this method twice with the same node.

void **addItem** (std::shared\_ptr<sh::filesystem::FilesystemNode> node) = 0

Adds an item to this list. In a model-backed list, this triggers the internal model mounting calls. It is not allowed to call this method twice with the same node.

void **removeItems** (QSet<std::shared\_ptr<sh::filesystem::FilesystemNode>> nodes) = 0

Removes nodes from this list. In a model-backed list, this triggers the internal model unmounting calls. It is not allowed to call this method with a node, which is not contained in that list.

void **removeItem** (std::shared\_ptr<sh::filesystem::FilesystemNode> node) = 0

Removes a node from this list. In a model-backed list, this triggers the internal model unmounting calls. It is not allowed to call this method with a node, which is not contained in that list.

void **resetItems** (sh::filesystem::FilesystemNodeType type, QSet<std::shared\_ptr<sh::filesystem::FilesystemNode>> nodes) = 0

Resets the list for a given node type to a given new list of children nodes.

std::shared\_ptr<sh::filesystem::FilesystemNode> **myNode** () = 0

Returns the node which owns this children list. The result may be 0 for non model-backed lists.

bool **contains** (std::shared\_ptr<FilesystemNode> node)

Checks if this list contains a given node.

const QList<std::shared\_ptr<sh::filesystem::FilesystemNode>> **\*nodes** ()

Returns the list of nodes currently stored in this instance.

**class FilesystemNodeListEditor**

*#include <filesystemnodelist.h>* A list editor for easily modifying a *FilesystemNodeList*.

Mainly used for listing child nodes in *sh::filesystem::FilesystemHandler.itemlist*. In most cases, you should just use *setItems*.

## Public Functions

**FilesystemNodeListEditor** (*FilesystemNodeList* \*list, std::shared\_ptr<const sh::filesystem::Eurl> parenteurl, sh::filesystem::FilesystemNodeType nodetype)

Constructed only by the infrastructure and made available otherwise.

**~FilesystemNodeListEditor** ()

void **setItems** (QList<QString> items)

Sets a new list content. This function is all you need in most situations. For iteratively adding nodes, which makes the intermediate results visible in the user interface, see *addItem*.

void **beginIterativeAdding** ()

Must be called before you begin to iteratively fill the children list (e.g. with *addItem*). You must also call *endIterativeAdding* afterwards.

void **addItem** (QString item)

Adds a new children to the list, directly showing that in the user interface, while you can proceed filling the list. Please read *beginIterativeAdding* as well! In most cases, you don't need this function.

```
std::shared_ptr<FilesystemNode> addCustomItem (std::shared_ptr<const
                                             sh::filesystem::Eurl> eurl,   QString
                                             displayname               =   QString(),
                                             sh::filesystem::FilesystemHandler *han-
                                             dler = 0)
```

Adds a new children to the list, directly showing that in the user interface, while you can proceed filling the list. Please read `beginIterativeAdding` as well! In most cases, you don't need this function.

Note: If you specify a handler, it must be the one matching to `eurl`'s scheme.

```
void addExistingNodeItem (std::shared_ptr<sh::filesystem::FilesystemNode> node)
```

Adds a new children to the list, directly showing that in the user interface, while you can proceed filling the list. Please read `beginIterativeAdding` as well! In most cases, you don't need this function.

```
void endIterativeAdding ()
```

Must be called after you iteratively filled the children list with `addItem`. This automatically removes all the old nodes, which you haven't added in this session.

```
void setItemsAreHosts ()
```

After this call, the `addItem` method will consider the given names as hostnames instead of new path segments. The resulting `sh::filesystem::Eurl` will differ accordingly. This only makes sense as children for root eurls.

```
FilesystemNodeList *rawlist ()
```

Returns the `FilesystemNodeList` backend. You should rarely need it.

## Private Functions

```
std::shared_ptr<const sh::filesystem::Eurl> getChildEurl (std::shared_ptr<const
                                                         sh::filesystem::Eurl> eurl,
                                                         QString item)
```

## Private Members

```
FilesystemNodeList *_list
```

```
std::shared_ptr<const sh::filesystem::Eurl> _parenteurl
```

```
sh::filesystem::FilesystemNodeType _nodetype
```

```
std::shared_ptr<sh::filesystem::FilesystemNode> _parentnode
```

```
sh::filesystem::FilesystemModel *_model
```

```
sh::filesystem::FilesystemHandler *_handler
```

```
bool _itemsAreHosts = false
```

```
bool _beganAddingIteratively = false
```

```
QSet<std::shared_ptr<sh::filesystem::FilesystemNode>> _iterativeAddingsBeforeState
```

### Private Static Attributes

QSet<FileSystemNodeList\*> **\_pendingIterativeAddings**

QMutex **\_pendingMutex**

QWaitCondition **\_iterativeAddingPossibleCondition**

### class FileSystemOperation

#include <filesystemoperation.h> A high-level interface for filesystem operations.

It is always based on the transaction of a *Operation* instance (which also gives you access to a *FileSystemOperation* object).

Some calls optionally allow to give an own instance of *sh::filesystem::FileSystemOperationProgressMonitor* for some additional functionality. Please note that not all calls provide all those functionality (some do not use e.g. the conflict resolution at all). If not provided, a default behavior is applied.

### Public Functions

**FileSystemOperation** (*sh::filesystem::Operation* \*operation)

Constructed only by the infrastructure and made available otherwise.

QList<std::shared\_ptr<const *sh::filesystem::Eurl*>> **itemlist** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*,  
*sh::filesystem::FileSystemNodeType*  
*type* =  
*sh::filesystem::FileSystemNodeType::NONE*)

Gets a list of shallot.sh::filesystem::FileSystemNode nodes in a directory (optionally filter by given types).

*sh::filesystem::FileSystemNodeType* **getType** (std::shared\_ptr<const *sh::filesystem::Eurl*>  
*eurl*)

Gets the shallot.sh::filesystem::FileSystemNodeType node type for an entry.

qint64 **getFileSize** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*)

Gets the file size for an entry.

QString **getLinkTarget** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*)

Gets the link target for an entry (if it is a link).

bool **canGetFileContent** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*)

Can we get the file content for an entry?

std::shared\_ptr<QIODevice> **getFileContent** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*)

Gets the file content for an entry as QIODevice.

bool **canCreateDirectory** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*)

Can we create a given directory?

void **createDirectory** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*,  
std::shared\_ptr<*sh::filesystem::FileSystemOperationProgressMonitor*>  
*progressmon* = 0)

Creates a directory.

#### Parameters

- *progressmon*: An optional progress monitor.

bool **canCreateLink** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*)

Can we create a given link?

void **createLink** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*, QString *target*,  
std::shared\_ptr<*sh::filesystem::FilesystemOperationProgressMonitor*>  
*progressmon* = 0)

Creates a link.

#### Parameters

- *progressmon*: An optional progress monitor.

bool **canCreateFile** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*)

Can we create a given file?

void **createFile** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*, QIODevice \**content*,  
std::shared\_ptr<*sh::filesystem::FilesystemOperationProgressMonitor*> *pro-*  
*gressmon* = 0)

Creates a file.

#### Parameters

- *progressmon*: An optional progress monitor.

bool **canDeleteItem** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*)

Can we delete a given entry?

void **deleteItem** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*,  
std::shared\_ptr<*sh::filesystem::FilesystemOperationProgressMonitor*>  
*progressmon* = 0)

Delete an entry.

#### Parameters

- *progressmon*: An optional progress monitor.

void **deleteDirectoryIfEmpty** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*,  
std::shared\_ptr<*sh::filesystem::FilesystemOperationProgressMonitor*>  
*progressmon* = 0)

Delete a directory entry if empty.

#### Parameters

- *progressmon*: An optional progress monitor.

bool **canMoveItem** (std::shared\_ptr<const *sh::filesystem::Eurl*> *src*, std::shared\_ptr<const  
*sh::filesystem::Eurl*> *dest* = 0)

Checks if it is allowed to move a certain item.

If the destination is known as well, it might help to pass it as well.

This does not guarantee success in the transfer, but is just a cheap early check.

void **moveItems** (QList<std::shared\_ptr<const *sh::filesystem::Eurl*>>  
*src*, std::shared\_ptr<const *sh::filesystem::Eurl*> *dest*,  
std::shared\_ptr<*sh::filesystem::FilesystemOperationProgressMonitor*>  
*progressmon* = 0)

Moves entries.

#### Parameters

- *dest*: The requested new common parent directory.
- *progressmon*: An optional progress monitor.

void **moveItem** (std::shared\_ptr<const *sh::filesystem::Eurl*> *src*,  
std::shared\_ptr<const *sh::filesystem::Eurl*> *dest*,  
std::shared\_ptr<*sh::filesystem::FilesystemOperationProgressMonitor*> *pro-*  
*gressmon* = 0)

Moves an entry.

#### Parameters

- `dest`: The new location (not the new parent).
- `progressmon`: An optional progress monitor.

bool **canCopyItem** (std::shared\_ptr<const *sh::filesystem::Eurl*> *src*, std::shared\_ptr<const *Eurl*> *dest* = 0)

Checks if it is allowed to copy a certain item.

If the destination is known as well, it might help to pass it as well.

This does not guarantee success in the transfer, but is just a cheap early check.

void **copyItems** (QList<std::shared\_ptr<const *sh::filesystem::Eurl*>>  
*src*, std::shared\_ptr<const *sh::filesystem::Eurl*> *dest*,  
std::shared\_ptr<*sh::filesystem::FilesystemOperationProgressMonitor*>  
*progressmon* = 0)

Copies entries.

#### Parameters

- `dest`: The requested new common parent directory.
- `progressmon`: An optional progress monitor.

void **copyItem** (std::shared\_ptr<const *sh::filesystem::Eurl*> *src*,  
std::shared\_ptr<const *sh::filesystem::Eurl*> *dest*,  
std::shared\_ptr<*sh::filesystem::FilesystemOperationProgressMonitor*> *pro-*  
*gressmon* = 0)

Copies an entry.

#### Parameters

- `dest`: The new location (not the new parent).
- `progressmon`: An optional progress monitor.

QList<std::shared\_ptr<*FilesystemNode*>> **resolveNodeLink** (std::shared\_ptr<*FilesystemNode*>  
*node*, bool *recursive* = true,  
bool *excludeOwn* = false)

Resolve a link as node with or without recursion.

std::shared\_ptr<const *sh::filesystem::Eurl*> **resolveEurlLink** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*,  
bool *recursive* = true, bool  
*excludeOwn* = false, int  
*tries* = 100)

Resolve a link as *sh::filesystem::Eurl* with or without recursion.

QList<QString> **listExtendedAttributes** (std::shared\_ptr<const *sh::filesystem::Eurl*>  
*eurl*)

quint64 **getExtendedAttributeSize** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*,  
QString *attribute*)

QByteArray **getExtendedAttribute** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*,  
QString *attribute*)

void **setExtendedAttribute** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*, QString  
*attribute*, QByteArray *value*)

void **removeExtendedAttribute** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*,  
QString *attribute*)

## Public Static Functions

void **addTransferrableDetailColumn** (int *index*, std::shared\_ptr<sh::filesystem::DetailColumn> *detailColumn*)

Registers a detail column for transferring it when file transfer take place.

### Parameters

- *index*: An integer which controls the order of transferring.

QList<std::shared\_ptr<sh::filesystem::DetailColumn>> **transferrableDetailColumns** ()

A list of all detail columns which are registered becoming transferred in file transfers.

## Private Functions

std::shared\_ptr<sh::filesystem::FilesystemHandler> **\_handler** (std::shared\_ptr<const sh::filesystem::Eurl> *eurl*)

std::shared\_ptr<sh::filesystem::FilesystemOperationProgressMonitor> **\_monitor** (std::shared\_ptr<sh::filesystem::FilesystemHandler> *h*)

std::shared\_ptr<const sh::filesystem::Eurl> **\_resolveIfNeeded** (std::shared\_ptr<sh::filesystem::FilesystemHandler> *handler*, std::shared\_ptr<const sh::filesystem::Eurl> *eurl*)

## Private Members

*Operation* \*\_**operation**

## Private Static Attributes

QMap<int, std::shared\_ptr<sh::filesystem::DetailColumn>> **\_detailColumnsMap**

QList<std::shared\_ptr<sh::filesystem::DetailColumn>> **\_detailColumns**

QMutex **\_detailColumnsMutex**

## Friends

**friend class** FilesystemOperationTransfers

**class** **MyHandlerTransfer** : **public** sh::filesystem::FilesystemHandler::HandlerTransfer

## Public Functions

**MyHandlerTransfer** (sh::filesystem::FilesystemOperationProgressMonitor \*\_progressmon = 0)

**~MyHandlerTransfer** ()

void **respectCancel** ()

void **incrementTransferredBytes** (qint64 *donebytes*)

## Private Members

*sh::filesystem::FilesystemOperationProgressMonitor* \*\_progressmon

**class FilesystemOperationProgressMonitor** : public std::enable\_shared\_from\_this<*FilesystemOperationProgressMonitor*>  
#include <filesystemoperationtransfers.h> Implement this class and use an instance of it as parameter in some methods of *sh::filesystem::FilesystemOperation* for some additional functionality.

This includes monitoring the progress, specifying a resolution when conflicts in the filesystem would occur and more.

It also makes sense to directly instantiate this class in some cases.

Subclassed by *sh::actions::common::ActionAbstractTransferTree::MyFilesystemOperationProgressMonitor*, *sh::scripting::api::ApiFilesystemOperationProgressMonitor*

## Public Functions

**FilesystemOperationProgressMonitor** (*sh::actions::ActionExecutionInfo* \*actionExecution = 0)

Is intended to be directly constructed from everywhere.

### Parameters

- actionExecution: An optional *sh::actions::ActionExecutionInfo* which is used in some places in the default implementation, when available.

bool **hasItemInfo** ()

Checks if this progress monitor provides information about how many items of a certain total number are transferred so far.

quint64 **doneItems** ()

Checks how many items are transferred so far.

quint64 **allItems** ()

Checks how many items are to be transferred in total (as predicted in the current moment).

bool **hasBytesInfo** ()

Checks if this progress monitor provides information about how many byte of a certain total number are transferred so far.

quint64 **doneBytes** ()

Checks how many bytes are transferred so far.

quint64 **allBytes** ()

Checks how many bytes are to be transferred in total (as predicted in the current moment).

QString **getItemInfoFrom** ()

Returns the current source of transfer (as textual information).

QString **getItemInfoTo** ()

Returns the current destination of transfer (as textual information).

QString **estimation** ()

Returns the current performance and time estimation (as textual information).

**~FilesystemOperationProgressMonitor** ()



## Private Functions

void **setProgress** (quint64 *doneitems*, quint64 *allitems*, quint64 *donebytes*, quint64 *allbytes*)  
 Used by *FilesystemOperationTransfers* for setting status changes.

void **incProgress** (quint64 *doneitems*, quint64 *allitems*, quint64 *donebytes*, quint64 *allbytes*)  
 Used by *FilesystemOperationTransfers* for setting status changes.

void **setItemInfo** (QString *from*, QString *to*)  
 Used by *FilesystemOperationTransfers* for setting status changes.

void **setEstimation** (QString *estimation*)  
 Used by *FilesystemOperationTransfers* for setting status changes.

void **\_enablestatistics** ()

void **\_computestatistics** ()

void **\_triggerchanged** ()

void **\_stoptriggerchanged** ()

## Private Members

QDateTime **\_laststatistictime**

quint64 **\_laststatisticdonebytes** = 0

quint64 **\_laststatisticdoneitems** = 0

double **\_statisticbytespeed** = 0.0

double **\_statisticitemspeed** = 0.0

int **\_statisticcountdown** = 4

QMutex **\_mutex**

QMutex **\_triggermutex**

quint64 **\_allitems** = 0

quint64 **\_doneitems** = 0

quint64 **\_allbytes** = 0

quint64 **\_donebytes** = 0

bool **\_triggerchanged\_changedrunning** = false

bool **\_triggerchanged\_changedrunagain** = false

bool **\_triggerchanged\_stopped** = false

QString **\_itemfrom**

QString **\_itemto**

QString **\_estimation**

QList<std::shared\_ptr<const *sh::filesystem::Eurl*>> **changes**

bool **\_begancomputestatistics** = false

## Friends

**friend class** FilesystemOperationTransfers

**friend class** FilesystemOperation

**class FilesystemOperationTransfers**

*#include <filesystemoperationtransfers.h>* Used by *FilesystemOperation* for managing the execution of transfer operations.

This class is only used directly by *FilesystemOperation* (but some inner class might be interesting).

## Public Functions

**FilesystemOperationTransfers** (*FilesystemOperation* *\*filesystem*,  
std::shared\_ptr<*FilesystemOperationProgressMonitor*>  
*progressmon*)

Constructed only by the infrastructure and made available otherwise.

void **executestepqueue** ()

Executes the queue.

The execution model is as follows: The complete queue of *OperationStepClass* steps becomes expanded, filesystem conflicts gets resolved and then all steps become executed.

In detail:

At first there is an interleaved process of expanding all steps (see *OperationStepClass::expand*) and checking/resolving filesystem conflicts in the fresh tree (see doreview). It checks all steps in the queue for conflicts (e.g. destination already exists). It expands each step which has no conflicts (by enqueueing it to the stepqueue). After those checks, it asks FilesystemOperationProgressMonitor::resolveConflicts for a resolution for all outstanding conflicts. With all the steps, which were (or are) in conflict, the entire process becomes repeated. This ends when all steps are expanded and no open conflicts exist anymore.

When this process is done, the execution phase begins. It iterates the queue. It takes the (in FIFO manner) first element (if one is there, otherwise stops). For that step, it makes a late-time conflict check at first. If a conflict appears, it becomes temporarily unexpanded. A resolution loop as above will run for this element, eventually leading to re-expansion. After the check, when no conflicts are open, it executes the step.

void **addMoveItem** (std::shared\_ptr<const *sh::filesystem::Eurl*> *src*, std::shared\_ptr<const *sh::filesystem::Eurl*> *dest*)

adds one move transfer into the queue.

void **addCopyItem** (std::shared\_ptr<const *sh::filesystem::Eurl*> *src*, std::shared\_ptr<const *sh::filesystem::Eurl*> *dest*)

adds one copy transfer into the queue.

**~FilesystemOperationTransfers** ()

## Private Functions

```
void _compile_moveItem (std::shared_ptr<const          sh::filesystem::Eurl>
                      src,          std::shared_ptr<const          sh::filesystem::Eurl>
                      dest,          QList<OperationStepClass*>    *result,
                      QList<OperationStepClass*> withExpansion)
```

Adds a move transfer step into given list.

```
void _compile_copyItem (std::shared_ptr<const          sh::filesystem::Eurl>
                      src,          std::shared_ptr<const          sh::filesystem::Eurl>
                      dest,          QList<OperationStepClass*>    *result,
                      QList<OperationStepClass*> withExpansion)
```

Adds a copy transfer step into given list.

```
void computeProgress ()
```

Computes the current progress (how many items/bytes are transferred from which total?) and notifies the progress monitor.

```
bool doreview (QList<OperationStepClass*> reviewsteps)
```

Makes checks if the given queue has conflicts and for asks `resolveConflicts` how to handle them.

For each non-conflicting step, it triggers its expansion, thereby modifying the stepqueue.

It returns when no open conflicts exist.

**Return** If there were any conflicts.

## Private Members

```
QList<OperationStepClass*> stepqueue
```

The queue of steps for execution.

```
OperationStepClass *currentstep = 0
```

The step which is currently in execution (or 0).

```
QList<OperationStepClass*> donesteps
```

Steps which were already executed.

```
FilesystemOperation *filesystem
```

```
std::shared_ptr<FilesystemOperationProgressMonitor> progressmon
```

```
class OperationStep
```

`#include <filesystemoperationtransfers.h>` One step of execution in a transfer operation by *FilesystemOperation*.

It is typically about a certain source and destination in the filesystem. It also can stay in conflict (due to checks from *FilesystemOperationTransfers::doreview*) and provides methods for resolving them (in *FilesystemOperationTransfers::FilesystemOperationProgressMonitor::resolveConflicts*).

Different subclasses implement distinct kinds of operations (e.g. copying or deleting).

Subclassed by *sh::filesystem::FilesystemOperationTransfers::OperationStepClass*

## Public Types

### **enum ConflictResolution**

Enumeration of ways how to resolve a filesystem conflict.

*Values:*

#### **enumerator Skip**

Skip this element.

#### **enumerator OverwriteDestination**

Overwrite the destination.

#### **enumerator RenameDestinationBefore**

Rename the file at the destination before transferring.

#### **enumerator UseDifferentDestinationName**

Transfer to another destination filename.

#### **enumerator MergeDirectories**

Merge source into destination directory (recursively).

#### **enumerator Unresolved**

No strategy.

#### **enumerator Indirect**

Indirectly solved. Only set by the engine.

## Public Functions

**OperationStep** (*FilesystemOperationTransfers* \*transfers, std::shared\_ptr<const  
sh::filesystem::Eurl> source, std::shared\_ptr<const  
sh::filesystem::Eurl> destination)

Constructed only by the infrastructure and made available otherwise.

QString **conflictDescription** ()

The description of the conflict (if any) as text.

*ConflictResolution* **conflictResolution** ()

The currently chosen conflict resolution.

QString **conflictResolution\_renameDestinationBeforeTo** ()

Returns new destination name (if current conflict resolution is *ConflictResolution::RenameDestinationBefore*).

QString **conflictResolution\_differentDestinationNameTo** ()

Returns new destination name (if current conflict resolution is *ConflictResolution::UseDifferentDestinationName*).

std::shared\_ptr<const sh::filesystem::Eurl> **source** ()

Returns the source location to be transferred (if specified).

std::shared\_ptr<const sh::filesystem::Eurl> **originalDestination** ()

Returns the destination location (if specified).

This is the complete target path, never the parent directory of the new element.

This is the original destination location. See also effectiveDestination.

std::shared\_ptr<const sh::filesystem::Eurl> **effectiveDestination** ()

Returns the real destination location with conflict resolution applied.

```

int sourcetype ()
    Returns the node type of the source.

    Should be overwritten in subclasses whenever it can determine the source type faster than the
    infrastructure.

void setConflictResolve_Skip ()
    Set conflict resolution to ConflictResolution::Skip.

void setConflictResolve_OverwriteDestination ()
    Set conflict resolution to ConflictResolution::OverwriteDestination.

void setConflictResolve_RenameDestinationBefore (QString newname)
    Set conflict resolution to ConflictResolution::RenameDestinationBefore.

void setConflictResolve_UseDifferentDestinationName (QString new-
                                                    name)
    Set conflict resolution to ConflictResolution::UseDifferentDestinationName.

void setConflictResolve_MergeDirectories ()
    Set conflict resolution to ConflictResolution::MergeDirectories.

    See also setConflictResolve_MergeDirectories_isAllowed.

bool setConflictResolve_MergeDirectories_checkAllowed ()
    Checks if conflict resolution ConflictResolution::MergeDirectories is allowed.

~OperationStep ()

```

## Friends

```

friend class FilesystemOperationTransfers

class OperationStep_ApplyDirectoryDetails : public sh::filesystem::FilesystemOperationTransfers:

```

## Public Types

```

enum ConflictResolution
    Enumeration of ways how to resolve a filesystem conflict.

    Values:

    enumerator Skip
        Skip this element.

    enumerator OverwriteDestination
        Overwrite the destination.

    enumerator RenameDestinationBefore
        Rename the file at the destination before transferring.

    enumerator UseDifferentDestinationName
        Transfer to another destination filename.

    enumerator MergeDirectories
        Merge source into destination directory (recursively).

    enumerator Unresolved
        No strategy.

    enumerator Indirect
        Indirectly solved. Only set by the engine.

```

## Public Functions

**OperationStep\_ApplyDirectoryDetails** (*FilesystemOperationTransfers*  
\*transfers, std::shared\_ptr<const  
sh::filesystem::Eurl> item,  
QList<QPair<sh::filesystem::DetailColumn\*,  
QVariant>> values,  
QList<OperationStepClass\*> with-  
Expansion)

void **execute** (*Operation* \*op)  
Implement this and handle the execution part of this step here.

bool **checkconflicts** (*Operation* \*op)  
Implement this for customizing the conflict checking.

QList<OperationStepClass\*> **expand** ()  
Implement this and handle the expansion part of this step here.

bool **hasOwnProgressIncreaseHandling** ()  
If it takes care on its own to signal increases in the transfer progress.

void **\_unexpand** (QList<OperationStepClass\*> \*intlist)  
Bookkeeping in *FilesystemOperationTransfers*.

void **\_expand** (QList<OperationStepClass\*> \*intlist)  
Bookkeeping in *FilesystemOperationTransfers*.

QList<QPair<sh::filesystem::DetailColumn\*, QVariant>> **getFileDetails** (sh::filesystem::Operation  
\*op,  
std::shared\_ptr<const  
sh::filesystem::Eurl>  
source)

void **applyFileDetails** (sh::filesystem::Operation \*op,  
QList<QPair<sh::filesystem::DetailColumn\*, QVariant>>  
values, std::shared\_ptr<const sh::filesystem::Eurl> destina-  
tion)

QString **conflictDescription** ()  
The description of the conflict (if any) as text.

*ConflictResolution* **conflictResolution** ()  
The currently chosen conflict resolution.

QString **conflictResolution\_renameDestinationBeforeTo** ()  
Returns new destination name (if current conflict resolution is *ConflictResolu-  
tion::RenameDestinationBefore*).

QString **conflictResolution\_differentDestinationNameTo** ()  
Returns new destination name (if current conflict resolution is *ConflictResolu-  
tion::UseDifferentDestinationName*).

std::shared\_ptr<const sh::filesystem::Eurl> **source** ()  
Returns the source location to be transferred (if specified).

std::shared\_ptr<const sh::filesystem::Eurl> **originalDestination** ()  
Returns the destination location (if specified).

This is the complete target path, never the parent directory of the new element.

This is the original destination location. See also effectiveDestination.

```
std::shared_ptr<const sh::filesystem::Eurl> effectiveDestination ()
    Returns the real destination location with conflict resolution applied.

int sourcetype ()
    Returns the node type of the source.

    Should be overwritten in subclasses whenever it can determine the source type faster than the
    infrastructure.

void setConflictResolve_Skip ()
    Set conflict resolution to ConflictResolution::Skip.

void setConflictResolve_OverwriteDestination ()
    Set conflict resolution to ConflictResolution::OverwriteDestination.

void setConflictResolve_RenameDestinationBefore (QString newname)
    Set conflict resolution to ConflictResolution::RenameDestinationBefore.

void setConflictResolve_UseDifferentDestinationName (QString new-
                                                    name)
    Set conflict resolution to ConflictResolution::UseDifferentDestinationName.

void setConflictResolve_MergeDirectories ()
    Set conflict resolution to ConflictResolution::MergeDirectories.

    See also setConflictResolve_MergeDirectories_isAllowed.

bool setConflictResolve_MergeDirectories_checkAllowed ()
    Checks if conflict resolution ConflictResolution::MergeDirectories is allowed.
```

## Public Members

```
QList<QPair<sh::filesystem::DetailColumn*, QVariant>> _detailvals

quint64 _cntItems = 0
    Number of items to be transferred in this step (used for progress monitoring).

quint64 _cntBytes = 0
    Number of byte to be transferred in this step (used for progress monitoring).

bool _isExpanded = false
    Bookkeeping in FilesystemOperationTransfers.

QList<OperationStepClass*> _expandnodes
    Bookkeeping in FilesystemOperationTransfers.

OperationStepClass *_parentnode = 0
    Bookkeeping in FilesystemOperationTransfers.

bool _deleteonunexpand = true
    Bookkeeping in FilesystemOperationTransfers.

QList<OperationStepClass*> _withExpansion

class OperationStep_CopyDirectory : public sh::filesystem::FilesystemOperationTransfers::OperationS
```

## Public Types

### **enum ConflictResolution**

Enumeration of ways how to resolve a filesystem conflict.

*Values:*

#### **enumerator Skip**

Skip this element.

#### **enumerator OverwriteDestination**

Overwrite the destination.

#### **enumerator RenameDestinationBefore**

Rename the file at the destination before transferring.

#### **enumerator UseDifferentDestinationName**

Transfer to another destination filename.

#### **enumerator MergeDirectories**

Merge source into destination directory (recursively).

#### **enumerator Unresolved**

No strategy.

#### **enumerator Indirect**

Indirectly solved. Only set by the engine.

## Public Functions

```
OperationStep_CopyDirectory (FilesystemOperationTransfers *transfers,  
                             std::shared_ptr<const sh::filesystem::Eurl>  
                             src, std::shared_ptr<const sh::filesystem::Eurl>  
                             dest, QList<OperationStepClass*> withExpansion)
```

```
void execute (Operation *op)
```

Implement this and handle the execution part of this step here.

```
QList<OperationStepClass*> expand ()
```

Implement this and handle the expansion part of this step here.

```
int sourcetype ()
```

Returns the node type of the source.

Should be overwritten in subclasses whenever it can determine the source type faster than the infrastructure.

```
bool checkconflicts (Operation *op)
```

Implement this for customizing the conflict checking.

```
bool hasOwnProgressIncreaseHandling ()
```

If it takes care on its own to signal increases in the transfer progress.

```
void _unexpand (QList<OperationStepClass*> *intlist)
```

Bookkeeping in *FilesystemOperationTransfers*.

```
void _expand (QList<OperationStepClass*> *intlist)
```

Bookkeeping in *FilesystemOperationTransfers*.



```

QList<QPair<sh::filesystem::DetailColumn*, QVariant>> getFileDetails (sh::filesystem::Operation
                                                                    *op,
                                                                    std::shared_ptr<const
                                                                    sh::filesystem::Eurl>
                                                                    source)

void applyFileDetails (sh::filesystem::Operation                      *op,
                      QList<QPair<sh::filesystem::DetailColumn*,    QVariant>>
                      values, std::shared_ptr<const sh::filesystem::Eurl> destina-
                      tion)

QString conflictDescription ()
    The description of the conflict (if any) as text.

ConflictResolution conflictResolution ()
    The currently chosen conflict resolution.

QString conflictResolution_renameDestinationBeforeTo ()
    Returns new destination name (if current conflict resolution is ConflictResolu-
    tion::RenameDestinationBefore).

QString conflictResolution_differentDestinationNameTo ()
    Returns new destination name (if current conflict resolution is ConflictResolu-
    tion::UseDifferentDestinationName).

std::shared_ptr<const sh::filesystem::Eurl> source ()
    Returns the source location to be transferred (if specified).

std::shared_ptr<const sh::filesystem::Eurl> originalDestination ()
    Returns the destination location (if specified).

    This is the complete target path, never the parent directory of the new element.

    This is the original destination location. See also effectiveDestination.

std::shared_ptr<const sh::filesystem::Eurl> effectiveDestination ()
    Returns the real destination location with conflict resolution applied.

void setConflictResolve_Skip ()
    Set conflict resolution to ConflictResolution::Skip.

void setConflictResolve_OverwriteDestination ()
    Set conflict resolution to ConflictResolution::OverwriteDestination.

void setConflictResolve_RenameDestinationBefore (QString newname)
    Set conflict resolution to ConflictResolution::RenameDestinationBefore.

void setConflictResolve_UseDifferentDestinationName (QString      new-
                                                         name)
    Set conflict resolution to ConflictResolution::UseDifferentDestinationName.

void setConflictResolve_MergeDirectories ()
    Set conflict resolution to ConflictResolution::MergeDirectories.

    See also setConflictResolve_MergeDirectories_isAllowed.

bool setConflictResolve_MergeDirectories_checkAllowed ()
    Checks if conflict resolution ConflictResolution::MergeDirectories is allowed.

```

## Public Members

quint64 **\_cntItems** = 0

Number of items to be transferred in this step (used for progress monitoring).

quint64 **\_cntBytes** = 0

Number of byte to be transferred in this step (used for progress monitoring).

bool **\_isExpanded** = false

Bookkeeping in *FilesystemOperationTransfers*.

QList<OperationStepClass\*> **\_expandnodes**

Bookkeeping in *FilesystemOperationTransfers*.

OperationStepClass \* **\_parentnode** = 0

Bookkeeping in *FilesystemOperationTransfers*.

bool **\_deleteonunexpand** = true

Bookkeeping in *FilesystemOperationTransfers*.

QList<OperationStepClass\*> **\_withExpansion**

**class** OperationStep\_CopyFile : public *sh::filesystem::FilesystemOperationTransfers::OperationStepClass*

## Public Types

**enum** ConflictResolution

Enumeration of ways how to resolve a filesystem conflict.

*Values:*

**enumerator** Skip

Skip this element.

**enumerator** OverwriteDestination

Overwrite the destination.

**enumerator** RenameDestinationBefore

Rename the file at the destination before transferring.

**enumerator** UseDifferentDestinationName

Transfer to another destination filename.

**enumerator** MergeDirectories

Merge source into destination directory (recursively).

**enumerator** Unresolved

No strategy.

**enumerator** Indirect

Indirectly solved. Only set by the engine.

## Public Functions

**OperationStep\_CopyFile** (*FilesystemOperationTransfers* *\*transfers*,  
 std::shared\_ptr<const *sh::filesystem::Eurl*> *src*,  
 std::shared\_ptr<const *sh::filesystem::Eurl*> *dest*,  
 QList<*OperationStepClass*> *withExpansion*)

void **execute** (*Operation* \**op*)  
 Implement this and handle the execution part of this step here.

int **sourcetype** ()  
 Returns the node type of the source.  
 Should be overwritten in subclasses whenever it can determine the source type faster than the infrastructure.

bool **hasOwnProgressIncreaseHandling** ()  
 If it takes care on its own to signal increases in the transfer progress.

QList<*OperationStepClass*> **expand** ()  
 Implement this and handle the expansion part of this step here.

bool **checkconflicts** (*Operation* \**op*)  
 Implement this for customizing the conflict checking.

void **\_unexpand** (QList<*OperationStepClass*> \**intlist*)  
 Bookkeeping in *FilesystemOperationTransfers*.

void **\_expand** (QList<*OperationStepClass*> \**intlist*)  
 Bookkeeping in *FilesystemOperationTransfers*.

QList<QPair<*sh::filesystem::DetailColumn*\*, QVariant>> **getFileDetails** (*sh::filesystem::Operation* \**op*,  
 std::shared\_ptr<const *sh::filesystem::Eurl*> *source*)

void **applyFileDetails** (*sh::filesystem::Operation* \**op*,  
 QList<QPair<*sh::filesystem::DetailColumn*\*, QVariant>> *values*, std::shared\_ptr<const *sh::filesystem::Eurl*> *destination*)

QString **conflictDescription** ()  
 The description of the conflict (if any) as text.

*ConflictResolution* **conflictResolution** ()  
 The currently chosen conflict resolution.

QString **conflictResolution\_renameDestinationBeforeTo** ()  
 Returns new destination name (if current conflict resolution is *ConflictResolution::RenameDestinationBefore*).

QString **conflictResolution\_differentDestinationNameTo** ()  
 Returns new destination name (if current conflict resolution is *ConflictResolution::UseDifferentDestinationName*).

std::shared\_ptr<const *sh::filesystem::Eurl*> **source** ()  
 Returns the source location to be transferred (if specified).

std::shared\_ptr<const *sh::filesystem::Eurl*> **originalDestination** ()  
 Returns the destination location (if specified).  
 This is the complete target path, never the parent directory of the new element.

This is the original destination location. See also `effectiveDestination`.

`std::shared_ptr<const sh::filesystem::Eurl> effectiveDestination ()`

Returns the real destination location with conflict resolution applied.

`void setConflictResolve_Skip ()`

Set conflict resolution to *ConflictResolution::Skip*.

`void setConflictResolve_OverwriteDestination ()`

Set conflict resolution to *ConflictResolution::OverwriteDestination*.

`void setConflictResolve_RenameDestinationBefore (QString newname)`

Set conflict resolution to *ConflictResolution::RenameDestinationBefore*.

`void setConflictResolve_UseDifferentDestinationName (QString new-name)`

Set conflict resolution to *ConflictResolution::UseDifferentDestinationName*.

`void setConflictResolve_MergeDirectories ()`

Set conflict resolution to *ConflictResolution::MergeDirectories*.

See also `setConflictResolve_MergeDirectories_isAllowed`.

`bool setConflictResolve_MergeDirectories_checkAllowed ()`

Checks if conflict resolution *ConflictResolution::MergeDirectories* is allowed.

## Public Members

`quint64 _cntItems = 0`

Number of items to be transferred in this step (used for progress monitoring).

`quint64 _cntBytes = 0`

Number of byte to be transferred in this step (used for progress monitoring).

`bool _isExpanded = false`

Bookkeeping in *FilesystemOperationTransfers*.

`QList<OperationStepClass*> _expandnodes`

Bookkeeping in *FilesystemOperationTransfers*.

`OperationStepClass *_parentnode = 0`

Bookkeeping in *FilesystemOperationTransfers*.

`bool _deleteonunexpand = true`

Bookkeeping in *FilesystemOperationTransfers*.

`QList<OperationStepClass*> _withExpansion`

`class OperationStep_CopyLink : public sh::filesystem::FilesystemOperationTransfers::OperationStepClass`

## Public Types

`enum ConflictResolution`

Enumeration of ways how to resolve a filesystem conflict.

*Values:*

`enumerator Skip`

Skip this element.

`enumerator OverwriteDestination`

Overwrite the destination.

**enumerator RenameDestinationBefore**  
 Rename the file at the destination before transferring.

**enumerator UseDifferentDestinationName**  
 Transfer to another destination filename.

**enumerator MergeDirectories**  
 Merge source into destination directory (recursively).

**enumerator Unresolved**  
 No strategy.

**enumerator Indirect**  
 Indirectly solved. Only set by the engine.

## Public Functions

**OperationStep\_CopyLink** (*FilesystemOperationTransfers* \*transfers,  
 std::shared\_ptr<const sh::filesystem::Eurl> src,  
 std::shared\_ptr<const sh::filesystem::Eurl> dest,  
 QList<OperationStepClass\*> withExpansion)

void **execute** (*Operation* \*op)  
 Implement this and handle the execution part of this step here.

int **sourcetype** ()  
 Returns the node type of the source.  
 Should be overwritten in subclasses whenever it can determine the source type faster than the infrastructure.

QList<OperationStepClass\*> **expand** ()  
 Implement this and handle the expansion part of this step here.

bool **checkconflicts** (*Operation* \*op)  
 Implement this for customizing the conflict checking.

bool **hasOwnProgressIncreaseHandling** ()  
 If it takes care on its own to signal increases in the transfer progress.

void **\_unexpand** (QList<OperationStepClass\*> \*intlist)  
 Bookkeeping in *FilesystemOperationTransfers*.

void **\_expand** (QList<OperationStepClass\*> \*intlist)  
 Bookkeeping in *FilesystemOperationTransfers*.

QList<QPair<sh::filesystem::DetailColumn\*, QVariant>> **getFileDetails** (sh::filesystem::Operation \*op,  
 std::shared\_ptr<const sh::filesystem::Eurl> source)

void **applyFileDetails** (sh::filesystem::Operation \*op,  
 QList<QPair<sh::filesystem::DetailColumn\*, QVariant>> values,  
 std::shared\_ptr<const sh::filesystem::Eurl> destination)

QString **conflictDescription** ()  
 The description of the conflict (if any) as text.

*ConflictResolution* **conflictResolution** ()  
 The currently chosen conflict resolution.

**QString conflictResolution\_renameDestinationBeforeTo()**  
Returns new destination name (if current conflict resolution is *ConflictResolution::RenameDestinationBefore*).

**QString conflictResolution\_differentDestinationNameTo()**  
Returns new destination name (if current conflict resolution is *ConflictResolution::UseDifferentDestinationName*).

**std::shared\_ptr<const sh::filesystem::Eurl> source()**  
Returns the source location to be transferred (if specified).

**std::shared\_ptr<const sh::filesystem::Eurl> originalDestination()**  
Returns the destination location (if specified).

This is the complete target path, never the parent directory of the new element.

This is the original destination location. See also *effectiveDestination*.

**std::shared\_ptr<const sh::filesystem::Eurl> effectiveDestination()**  
Returns the real destination location with conflict resolution applied.

**void setConflictResolve\_Skip()**  
Set conflict resolution to *ConflictResolution::Skip*.

**void setConflictResolve\_OverwriteDestination()**  
Set conflict resolution to *ConflictResolution::OverwriteDestination*.

**void setConflictResolve\_RenameDestinationBefore(QString newname)**  
Set conflict resolution to *ConflictResolution::RenameDestinationBefore*.

**void setConflictResolve\_UseDifferentDestinationName(QString newname)**  
Set conflict resolution to *ConflictResolution::UseDifferentDestinationName*.

**void setConflictResolve\_MergeDirectories()**  
Set conflict resolution to *ConflictResolution::MergeDirectories*.

See also *setConflictResolve\_MergeDirectories\_isAllowed*.

**bool setConflictResolve\_MergeDirectories\_checkAllowed()**  
Checks if conflict resolution *ConflictResolution::MergeDirectories* is allowed.

## Public Members

**quint64 \_cntItems = 0**  
Number of items to be transferred in this step (used for progress monitoring).

**quint64 \_cntBytes = 0**  
Number of byte to be transferred in this step (used for progress monitoring).

**bool \_isExpanded = false**  
Bookkeeping in *FilesystemOperationTransfers*.

**QList<OperationStepClass\*> \_expandnodes**  
Bookkeeping in *FilesystemOperationTransfers*.

**OperationStepClass \*\_parentnode = 0**  
Bookkeeping in *FilesystemOperationTransfers*.

**bool \_deleteonunexpand = true**  
Bookkeeping in *FilesystemOperationTransfers*.

**QList<OperationStepClass\*> \_withExpansion**

```
class OperationStep_DeleteItem: public sh::filesystem::FilesystemOperationTransfers::OperationStepC
```

## Public Types

### enum ConflictResolution

Enumeration of ways how to resolve a filesystem conflict.

*Values:*

#### enumerator Skip

Skip this element.

#### enumerator OverwriteDestination

Overwrite the destination.

#### enumerator RenameDestinationBefore

Rename the file at the destination before transferring.

#### enumerator UseDifferentDestinationName

Transfer to another destination filename.

#### enumerator MergeDirectories

Merge source into destination directory (recursively).

#### enumerator Unresolved

No strategy.

#### enumerator Indirect

Indirectly solved. Only set by the engine.

## Public Functions

```
OperationStep_DeleteItem (FilesystemOperationTransfers *transfers,
                          std::shared_ptr<const sh::filesystem::Eurl> src,
                          std::shared_ptr<const sh::filesystem::Eurl> dest,
                          QList<OperationStepClass*> withExpansion)
```

```
void execute (Operation *op)
```

Implement this and handle the execution part of this step here.

```
bool checkconflicts (Operation *op)
```

Implement this for customizing the conflict checking.

```
int sourcetype ()
```

Returns the node type of the source.

Should be overwritten in subclasses whenever it can determine the source type faster than the infrastructure.

```
QList<OperationStepClass*> expand ()
```

Implement this and handle the expansion part of this step here.

```
bool hasOwnProgressIncreaseHandling ()
```

If it takes care on its own to signal increases in the transfer progress.

```
void _unexpand (QList<OperationStepClass*> *intlist)
```

Bookkeeping in *FilesystemOperationTransfers*.

```
void _expand (QList<OperationStepClass*> *intlist)
```

Bookkeeping in *FilesystemOperationTransfers*.

```
QList<QPair<sh::filesystem::DetailColumn*, QVariant>> getFileDetails (sh::filesystem::Operation
                                                                    *op,
                                                                    std::shared_ptr<const
                                                                    sh::filesystem::Eurl>
                                                                    source)

void applyFileDetails (sh::filesystem::Operation                                *op,
                      QList<QPair<sh::filesystem::DetailColumn*,      QVariant>>
                      values, std::shared_ptr<const sh::filesystem::Eurl> destina-
                      tion)

QString conflictDescription ()
    The description of the conflict (if any) as text.

ConflictResolution conflictResolution ()
    The currently chosen conflict resolution.

QString conflictResolution_renameDestinationBeforeTo ()
    Returns new destination name (if current conflict resolution is ConflictResolu-
    tion::RenameDestinationBefore).

QString conflictResolution_differentDestinationNameTo ()
    Returns new destination name (if current conflict resolution is ConflictResolu-
    tion::UseDifferentDestinationName).

std::shared_ptr<const sh::filesystem::Eurl> source ()
    Returns the source location to be transferred (if specified).

std::shared_ptr<const sh::filesystem::Eurl> originalDestination ()
    Returns the destination location (if specified).

    This is the complete target path, never the parent directory of the new element.

    This is the original destination location. See also effectiveDestination.

std::shared_ptr<const sh::filesystem::Eurl> effectiveDestination ()
    Returns the real destination location with conflict resolution applied.

void setConflictResolve_Skip ()
    Set conflict resolution to ConflictResolution::Skip.

void setConflictResolve_OverwriteDestination ()
    Set conflict resolution to ConflictResolution::OverwriteDestination.

void setConflictResolve_RenameDestinationBefore (QString newname)
    Set conflict resolution to ConflictResolution::RenameDestinationBefore.

void setConflictResolve_UseDifferentDestinationName (QString          new-
                                                         name)
    Set conflict resolution to ConflictResolution::UseDifferentDestinationName.

void setConflictResolve_MergeDirectories ()
    Set conflict resolution to ConflictResolution::MergeDirectories.

    See also setConflictResolve_MergeDirectories_isAllowed.

bool setConflictResolve_MergeDirectories_checkAllowed ()
    Checks if conflict resolution ConflictResolution::MergeDirectories is allowed.
```



## Public Members

quint64 **\_cntItems** = 0

Number of items to be transferred in this step (used for progress monitoring).

quint64 **\_cntBytes** = 0

Number of byte to be transferred in this step (used for progress monitoring).

bool **\_isExpanded** = false

Bookkeeping in *FilesystemOperationTransfers*.

QList<*OperationStepClass*> **\_expandnodes**

Bookkeeping in *FilesystemOperationTransfers*.

*OperationStepClass* \* **\_parentnode** = 0

Bookkeeping in *FilesystemOperationTransfers*.

bool **\_deleteonunexpand** = true

Bookkeeping in *FilesystemOperationTransfers*.

QList<*OperationStepClass*> **\_withExpansion**

**class** **OperationStep\_RenameItem**: **public** *sh::filesystem::FilesystemOperationTransfers::OperationStepC*

## Public Types

**enum** **ConflictResolution**

Enumeration of ways how to resolve a filesystem conflict.

*Values:*

**enumerator** **Skip**

Skip this element.

**enumerator** **OverwriteDestination**

Overwrite the destination.

**enumerator** **RenameDestinationBefore**

Rename the file at the destination before transferring.

**enumerator** **UseDifferentDestinationName**

Transfer to another destination filename.

**enumerator** **MergeDirectories**

Merge source into destination directory (recursively).

**enumerator** **Unresolved**

No strategy.

**enumerator** **Indirect**

Indirectly solved. Only set by the engine.

## Public Functions

**OperationStep\_RenameItem** (*FilesystemOperationTransfers* *\*transfers*,  
std::shared\_ptr<const *sh::filesystem::Eurl*> *src*,  
QString *destpath*, QList<*OperationStepClass\**> *withExpansion*)

QList<*OperationStepClass\**> **expand** ()  
Implement this and handle the expansion part of this step here.

void **execute** (*Operation* *\*op*)  
Implement this and handle the execution part of this step here.

bool **checkconflicts** (*Operation* *\*op*)  
Implement this for customizing the conflict checking.

bool **hasOwnProgressIncreaseHandling** ()  
If it takes care on its own to signal increases in the transfer progress.

void **\_unexpand** (QList<*OperationStepClass\**> *\*intlist*)  
Bookkeeping in *FilesystemOperationTransfers*.

void **\_expand** (QList<*OperationStepClass\**> *\*intlist*)  
Bookkeeping in *FilesystemOperationTransfers*.

QList<QPair<*sh::filesystem::DetailColumn\**, QVariant>> **getFileDetails** (*sh::filesystem::Operation*  
*\*op*,  
std::shared\_ptr<const *sh::filesystem::Eurl*>  
*source*)

void **applyFileDetails** (*sh::filesystem::Operation* *\*op*,  
QList<QPair<*sh::filesystem::DetailColumn\**, QVariant>>  
*values*, std::shared\_ptr<const *sh::filesystem::Eurl*> *destination*)

QString **conflictDescription** ()  
The description of the conflict (if any) as text.

*ConflictResolution* **conflictResolution** ()  
The currently chosen conflict resolution.

QString **conflictResolution\_renameDestinationBeforeTo** ()  
Returns new destination name (if current conflict resolution is *ConflictResolution::RenameDestinationBefore*).

QString **conflictResolution\_differentDestinationNameTo** ()  
Returns new destination name (if current conflict resolution is *ConflictResolution::UseDifferentDestinationName*).

std::shared\_ptr<const *sh::filesystem::Eurl*> **source** ()  
Returns the source location to be transferred (if specified).

std::shared\_ptr<const *sh::filesystem::Eurl*> **originalDestination** ()  
Returns the destination location (if specified).

This is the complete target path, never the parent directory of the new element.

This is the original destination location. See also *effectiveDestination*.

std::shared\_ptr<const *sh::filesystem::Eurl*> **effectiveDestination** ()  
Returns the real destination location with conflict resolution applied.

**int sourcetype ()**  
Returns the node type of the source.

Should be overwritten in subclasses whenever it can determine the source type faster than the infrastructure.

**void setConflictResolve\_Skip ()**  
Set conflict resolution to *ConflictResolution::Skip*.

**void setConflictResolve\_OverwriteDestination ()**  
Set conflict resolution to *ConflictResolution::OverwriteDestination*.

**void setConflictResolve\_RenameDestinationBefore (QString newname)**  
Set conflict resolution to *ConflictResolution::RenameDestinationBefore*.

**void setConflictResolve\_UseDifferentDestinationName (QString new-name)**  
Set conflict resolution to *ConflictResolution::UseDifferentDestinationName*.

**void setConflictResolve\_MergeDirectories ()**  
Set conflict resolution to *ConflictResolution::MergeDirectories*.

See also `setConflictResolve_MergeDirectories_isAllowed`.

**bool setConflictResolve\_MergeDirectories\_checkAllowed ()**  
Checks if conflict resolution *ConflictResolution::MergeDirectories* is allowed.

## Public Members

**quint64 \_cntItems = 0**  
Number of items to be transferred in this step (used for progress monitoring).

**quint64 \_cntBytes = 0**  
Number of byte to be transferred in this step (used for progress monitoring).

**bool \_isExpanded = false**  
Bookkeeping in *FilesystemOperationTransfers*.

**QList<OperationStepClass\*> \_expandnodes**  
Bookkeeping in *FilesystemOperationTransfers*.

**OperationStepClass \*\_parentnode = 0**  
Bookkeeping in *FilesystemOperationTransfers*.

**bool \_deleteonunexpand = true**  
Bookkeeping in *FilesystemOperationTransfers*.

**QList<OperationStepClass\*> \_withExpansion**

**class OperationStepClass : public sh::filesystem::FilesystemOperationTransfers::OperationStep**  
A base class for all implementations.

They are used from inside *FilesystemOperationTransfers::executestepqueue*.

Subclassed by *sh::filesystem::FilesystemOperationTransfers::OperationStep\_ApplyDirectoryDetails*,  
*sh::filesystem::FilesystemOperationTransfers::OperationStep\_CopyDirectory*,  
*sh::filesystem::FilesystemOperationTransfers::OperationStep\_CopyFile*,  
*sh::filesystem::FilesystemOperationTransfers::OperationStep\_CopyLink*,  
*sh::filesystem::FilesystemOperationTransfers::OperationStep\_DeleteItem*,  
*sh::filesystem::FilesystemOperationTransfers::OperationStep\_RenameItem*

## Public Types

### **enum ConflictResolution**

Enumeration of ways how to resolve a filesystem conflict.

*Values:*

#### **enumerator Skip**

Skip this element.

#### **enumerator OverwriteDestination**

Overwrite the destination.

#### **enumerator RenameDestinationBefore**

Rename the file at the destination before transferring.

#### **enumerator UseDifferentDestinationName**

Transfer to another destination filename.

#### **enumerator MergeDirectories**

Merge source into destination directory (recursively).

#### **enumerator Unresolved**

No strategy.

#### **enumerator Indirect**

Indirectly solved. Only set by the engine.

## Public Functions

**OperationStepClass** (*FilesystemOperationTransfers* \*transfers, std::shared\_ptr<const sh::filesystem::Eurl> source, std::shared\_ptr<const Eurl> destination, QList<OperationStepClass\*> withExpansion)

void **execute** (*Operation* \*op) = 0

Implement this and handle the execution part of this step here.

QList<OperationStepClass\*> **expand** ()

Implement this and handle the expansion part of this step here.

bool **checkconflicts** (*Operation* \*op)

Implement this for customizing the conflict checking.

bool **hasOwnProgressIncreaseHandling** ()

If it takes care on its own to signal increases in the transfer progress.

void **\_unexpand** (QList<OperationStepClass\*> \*intlist)

Bookkeeping in *FilesystemOperationTransfers*.

void **\_expand** (QList<OperationStepClass\*> \*intlist)

Bookkeeping in *FilesystemOperationTransfers*.

**~OperationStepClass** ()

QList<QPair<sh::filesystem::DetailColumn\*, QVariant>> **getFileDetails** (sh::filesystem::Operation \*op, std::shared\_ptr<const sh::filesystem::Eurl> source)

```

void applyFileDetails (sh::filesystem::Operation *op,
                      QList<QPair<sh::filesystem::DetailColumn*, QVariant>>
                      values, std::shared_ptr<const sh::filesystem::Eurl> destina-
                      tion)

QString conflictDescription ()
    The description of the conflict (if any) as text.

ConflictResolution conflictResolution ()
    The currently chosen conflict resolution.

QString conflictResolution_renameDestinationBeforeTo ()
    Returns new destination name (if current conflict resolution is ConflictResolu-
    tion::RenameDestinationBefore).

QString conflictResolution_differentDestinationNameTo ()
    Returns new destination name (if current conflict resolution is ConflictResolu-
    tion::UseDifferentDestinationName).

std::shared_ptr<const sh::filesystem::Eurl> source ()
    Returns the source location to be transferred (if specified).

std::shared_ptr<const sh::filesystem::Eurl> originalDestination ()
    Returns the destination location (if specified).

    This is the complete target path, never the parent directory of the new element.

    This is the original destination location. See also effectiveDestination.

std::shared_ptr<const sh::filesystem::Eurl> effectiveDestination ()
    Returns the real destination location with conflict resolution applied.

int sourcetype ()
    Returns the node type of the source.

    Should be overwritten in subclasses whenever it can determine the source type faster than the
    infrastructure.

void setConflictResolve_Skip ()
    Set conflict resolution to ConflictResolution::Skip.

void setConflictResolve_OverwriteDestination ()
    Set conflict resolution to ConflictResolution::OverwriteDestination.

void setConflictResolve_RenameDestinationBefore (QString newname)
    Set conflict resolution to ConflictResolution::RenameDestinationBefore.

void setConflictResolve_UseDifferentDestinationName (QString new-
                                                    name)
    Set conflict resolution to ConflictResolution::UseDifferentDestinationName.

void setConflictResolve_MergeDirectories ()
    Set conflict resolution to ConflictResolution::MergeDirectories.

    See also setConflictResolve_MergeDirectories_isAllowed.

bool setConflictResolve_MergeDirectories_checkAllowed ()
    Checks if conflict resolution ConflictResolution::MergeDirectories is allowed.

```

## Public Members

quint64 **\_cntItems** = 0

Number of items to be transferred in this step (used for progress monitoring).

quint64 **\_cntBytes** = 0

Number of byte to be transferred in this step (used for progress monitoring).

bool **\_isExpanded** = false

Bookkeeping in *FilesystemOperationTransfers*.

QList<*OperationStepClass*> **\_expandnodes**

Bookkeeping in *FilesystemOperationTransfers*.

*OperationStepClass* \* **\_parentnode** = 0

Bookkeeping in *FilesystemOperationTransfers*.

bool **\_deleteonunexpand** = true

Bookkeeping in *FilesystemOperationTransfers*.

QList<*OperationStepClass*> **\_withExpansion**

**class SingleStepMonitor**: public *sh::filesystem::FilesystemHandler::HandlerTransfer*

Used for managing detailed progress changes about a single step.

## Public Functions

**SingleStepMonitor** (*OperationStepClass* \*step)

void **respectCancel** ()

void **incrementTransferredBytes** (quint64 donebytes)

**~SingleStepMonitor** ()

## Private Members

*OperationStepClass* \***step**

quint64 **\_donebytes** = 0

**class LoadOnDemandPlaceholderFilesystemNode**: public *sh::filesystem::FilesystemNode*

*#include <filesystemnode.h>* A special node, which is unselectable and shows a 'loading' label.

It does not correspond to a file, directory or similar which actually exists. It is also never parent or a child of another node.

They are used internally by the filesystem model.

## Public Functions

QString **displayName** ()

Returns the display name (what the user interface displays).

void **requestDetails** (bool *force*)

Requests to fetch details for this node.

**LoadOnDemandPlaceholderFilesystemNode** (std::shared\_ptr<sh::filesystem::FilesystemNode>  
parent)

Constructed only by the infrastructure and made available otherwise.

std::shared\_ptr<const sh::filesystem::Eurl> **eurl** ()

Returns the *sh::filesystem::Eurl* entry address.

sh::filesystem::FilesystemModel \***model** ()

Convenience shortcut to the *sh::filesystem::FilesystemModel*.

void **setDisplayName** (QString *displayname*)

Sets the display name (what the user interface displays).

sh::filesystem::FilesystemHandler \***handler** ()

Returns the handler which is responsible for this node. This should be the same as *sh::filesystem::FilesystemHandlerRegister.findHandler(eurl.scheme)*.

int **nodetype** ()

Returns the FilesystemNodeType node type.

int **childnodeCount** ()

Returns the number of children nodes.

std::shared\_ptr<sh::filesystem::FilesystemNode> **childnode** (int *i*)

Returns i-th child node.

sh::filesystem::ModelBackedFilesystemNodeList \***childnodes** ()

Returns the list of children.

std::shared\_ptr<sh::filesystem::FilesystemNode> **parentnode** () const

Returns the parent node.

This is 0 if the node does not exist in the model (i.e. not yet inserted or removed afterwards). In most cases, this can be interpreted as ‘file currently does not exist’.

int **index** ()

Returns the index of this node in the parent’s list of children.

void **addChild** (std::shared\_ptr<sh::filesystem::FilesystemNode> *node*)

Adds a new child. Not allowed to call more than once for a node.

void **removeChild** (std::shared\_ptr<sh::filesystem::FilesystemNode> *node*)

Removes a new child. Not allowed to call more than once for a node.

bool **isFetchingSubitems** ()

Checks if currently subitems are fetched (i.e. the ‘loading’ node is shown).

QIcon **icon** ()

Returns the node icon.

void **setIcon** (QIcon *icon*)

Sets the node icon.

bool **isHidden** ()

Returns if the node is hidden.

Note: It's not difficult for the user to also show the hidden items.

void **setHidden** (bool *v*)

Sets if the node is hidden.

See also *isHidden()*.

void **addDetail** (std::shared\_ptr<*sh::filesystem::DetailColumn*> *column*)

Adds a detail to this node.

std::shared\_ptr<*sh::filesystem::DetailColumn*> **getDetailColumnByIndex** (int *index*)

Returns the *i*-th detail column registered to this node.

int **getDetailColumnsCount** () const

Returns the number of detail columns registered to this node.

bool **isRootNode** () const

Checks if this is the root node of the model.

bool **isConfigured** () const

Checks if this node was already configured by a handler.

bool **isAlive** () const

Checks if this node currently exists in the model.

QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> **linkDestinationNodes** ()

Returns the list of link source nodes. If this node is a link, the link destination nodes may influence the behavior and appearance of this node.

This information does not come from inside but must be set from outside the model implementation.

void **setLinkDestinationNodes** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>>  
*linkdestinations*)

Sets the list of link destination nodes. If this node is a link, the link destination nodes may influence the behavior and appearance of this node.

void **requestContainedItems** (*sh::filesystem::FilesystemNodeType* *type* = *FilesystemNodeType::NONE*)

Requests to fetch the children of this node with help of the filesystem handler.

## Signals

void **removed** ()

Is emitted when this node is not placed in the tree anymore. This does not mean the physical deletion of the instance, but just means it is not alive from now on.

void **\_detailsAvailable** ()

Is emitted when values for detail columns arrived.

void **iconChanged** ()

Is emitted when the icon changed.

void **isHiddenChanged** ()

Is emitted when the hidden flag changed.



## Private Members

bool **loading**

## Friends

**friend class** *sh::filesystem::FilesystemModel*

**class** **ModelBackedFilesystemNodeList** : **public** *sh::filesystem::FilesystemNodeList*  
*#include <filesystemmodelist.h>* This *FilesystemNodeList* subclass builds a part of the filesystem model.

The listed nodes are actually owned and handled by a *sh::filesystem::FilesystemNode*. Node additions and removals will directly influence the model.

## Public Functions

**ModelBackedFilesystemNodeList** (*sh::filesystem::FilesystemNode \*node*)

Constructed only internally.

void **addItem** (QSet<std::shared\_ptr<*FilesystemNode*>> *nodes*)

void **addItem** (std::shared\_ptr<*FilesystemNode*> *node*)

void **removeItems** (QSet<std::shared\_ptr<*FilesystemNode*>> *nodes*)

void **removeItem** (std::shared\_ptr<*FilesystemNode*> *node*)

void **resetItems** (*sh::filesystem::FilesystemNodeType* *type*, QSet<std::shared\_ptr<*FilesystemNode*>> *nodes*)

std::shared\_ptr<*FilesystemNode*> **myNode** ()

Returns the node which owns this children list. The result may be 0 for non model-backed lists.

void **clear** ()

Clears the entire list. This is not a high-level operation, but something only used internally (not part of the interface).

void **addPermanentItem** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> *node*)

Adds an item to this list. In a model-backed list, this triggers the internal model mounting calls. It is not allowed to call this method twice with the same node.

This list is safe against removing from later item lists. Only **removePermanentItem** removes it. See also **addItem**.

void **removePermanentItem** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> *node*)

Removes a permanent node from this list. In a model-backed list, this triggers the internal model unmounting calls. It is not allowed to call this method with a node, which is not contained in that list.

void **addItem** (QSet<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> *nodes*) = 0

Adds items to this list. In a model-backed list, this triggers the internal model mounting calls. It is not allowed to call this method twice with the same node.

void **addItem** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> *node*) = 0

Adds an item to this list. In a model-backed list, this triggers the internal model mounting calls. It is not allowed to call this method twice with the same node.

```
void removeItems (QSet<std::shared_ptr<sh::filesystem::FilesystemNode>> nodes) = 0
    Removes nodes from this list. In a model-backed list, this triggers the internal model unmounting
    calls. It is not allowed to call this method with a node, which is not contained in that list.

void removeItem (std::shared_ptr<sh::filesystem::FilesystemNode> node) = 0
    Removes a node from this list. In a model-backed list, this triggers the internal model unmounting
    calls. It is not allowed to call this method with a node, which is not contained in that list.

void resetItems (sh::filesystem::FilesystemNodeType type, QSet<std::shared_ptr<sh::filesystem::FilesystemNode>>
                nodes) = 0
    Resets the list for a given node type to a given new list of children nodes.

bool contains (std::shared_ptr<FilesystemNode> node)
    Checks if this list contains a given node.

const QList<std::shared_ptr<sh::filesystem::FilesystemNode>> *nodes ()
    Returns the list of nodes currently stored in this instance.
```

### Private Functions

```
void addItem_helper (std::shared_ptr<sh::filesystem::FilesystemNode> node)
```

### Private Members

```
QSet<sh::filesystem::FilesystemNode*> _permanentnodes
sh::filesystem::FilesystemNode *_mynode
sh::filesystem::FilesystemModel *mymodel
```

```
class Operation : public QObject
```

```
#include <operation.h> A operation surrounds a completed series of steps in the filesystem.
```

For some steps to execute in the filesystem (i.e. not only the local one but the entire *Eurl* universe), it fetches intermediate files, caches them and writes them back to their real location afterwards.

It provides transaction-like operations for committing/dropping those intermediate files. A high-level interface for file management is available in *filesystem*.

It is allowed to create new instances from scratch, but you should check if you implicitly have an instance available in your situation, or if *sh::tools::OperationsCache* is an option.

### Public Functions

```
Operation (QObject *parent = 0)
```

Often constructed by the infrastructure and made available, but can also be constructed directly.

```
quint64 getFreeCacheSpace ()
```

Returns the free disk space (in bytes) available for operations like *fetchContainerFile()* or *fetchFile()*.

```
QString fetchContainerFile (std::shared_ptr<const sh::filesystem::Eurl> eurl, QString
                           namehint)
```

Fetches the container file for a eurl locally and returns the local path. Use this with care and only if needed. In many cases working with streams is the better way!

```
QString fetchContainerFile (std::shared_ptr<const sh::filesystem::Eurl> eurl)
```

QString **fetchFile** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*, QString *namehint*)  
Fetches a file locally and returns the local path. Use this with care and only if needed. In many cases working with streams is the better way!

QString **fetchFile** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*)

void **abort** ()  
Aborts transaction dropping all pending changes.

void **commit** ()  
Commits transaction applying all pending changes.

void **enableDetailsCache** ()  
Enable details caching.

bool **isDetailsCacheEnabled** ()  
Is details caching enabled?

void **storeDetailInCache** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*,  
*sh::filesystem::DetailColumn* \**column*, QVariant *value*)  
Stores a column detail in cache.

QVariant **getDetailFromCache** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*, const  
*sh::filesystem::DetailColumn* \**column*)  
Gets a column value from cache.

QList<std::shared\_ptr<const *sh::filesystem::Eurl*>> **pendingCommits** ()  
Returns a list of *sh::filesystem::Eurl* items which are marked for commit.

QString **getTempDir** ()  
Creates a fresh temporary folder and returns the path to it.

void **setCustomData** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*, QString *key*, QVariant  
*data*)  
Stores custom data.

QVariant **getCustomData** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*, QString *key*)  
Gets stored custom data.

void **setMaxAllowedSizeRatioPerPart** (double *v*)  
Sets a maximum part of available disk space for usage. This is exotic functionality you typically don't need. .

#### Parameters

- *v*: A ratio (between 0 and 1) of the available disk space which one part maximally may cost.

*sh::filesystem::FilesystemOperation* \***filesystem** ()  
Gets the *sh::filesystem::FilesystemOperation* filesystem operation object.

**~Operation** ()

## Public Static Functions

void **writeIODeviceToFile** (QIODevice \**content*, QString *filepath*)  
Low-level function for writing a QIODevice content to a local path.

### Private Members

```
QHash<std::shared_ptr<const sh::filesystem::Eurl>, QString> files
QSet<std::shared_ptr<const sh::filesystem::Eurl>> fileIsTemporary
QList<QString> tempDirs
bool _detailsCacheEnabled = false
QHash<std::shared_ptr<const sh::filesystem::Eurl>, QMap<QString, QVariant>> _customData
QHash<const sh::filesystem::DetailColumn*, QHash<std::shared_ptr<const sh::filesystem::Eurl>, QVariant>> _de
sh::filesystem::FilesystemOperation *_filesystemOperation
double _maxAllowedSizeRatioPerPart
QMutex operationlock
```

### Private Static Attributes

```
QMutex _tempfilemutex
QString _tempdir
class MaxAllowedSizeRatioPerPartExceededException : public sh::exceptions::IOException
    #include <operation.h> IO exception raised when the value in setMaxAllowedSizeRatioPerPart()
    was exceeded.
```

### Public Functions

```
MaxAllowedSizeRatioPerPartExceededException ()
QString message () const
QString name () const
QString classes () const
QString details () const
QString callstack () const
QString auxiliary () const
QString customValue (QString key) const
bool isRuntimeException () const
bool isProgramException () const
bool isRetryable () const
bool isResumeable () const
bool isDetailsAreInteresting () const
int autoRetryRecommendedIn () const
void setResumeable (bool v)
void setReTryable (bool v)
void setCustomValue (QString key, QString value)
```

```
bool isClass (QString classname)
sh::exceptions::ExceptionData data ()
```

## Public Static Functions

```
template<class Handler>
std::shared_ptr<RegisterHandler<Handler>> createRegisterHandler (Handler
                                                                    handler,  QS-
                                                                    tack<Handler>
                                                                    *stack)
```

```
void executeGuarded (std::function<void>
    > fc int flags = 0) Executes some code with some standard exection handling around.
```

### Parameters

- *fc*: The code to execute guarded.
- *flags*: Flags of *ExecuteGuardFlag* for choosing the behavior.

```
void executeGuarded_errorpanel (std::function<void>
    > fc int flags = 0)
```

```
QString _demangle (QString l)
```

```
void exceptionDialog (sh::exceptions::Exception &e, bool *doRetry)
```

## Public Static Attributes

```
const QString UNDEFINED_ERROR_OCCURRED = QObject::tr("An unspecified error occurred.")
```

```
const QString SHALLOT_MUST_CLOSE_TEXT = QObject::tr("Your Shallot process is disturbed by an error and
```

```
const QString Value_isShallotException = "_isShallotException"
```

## namespace filesystemhandlers

Implementations of filesystem handlers.

Subclasses of *sh::filesystem::FilesystemHandler* (and possibly some auxiliary stuff). They implement how to access various types of filesystems.

Some special purpose handlers reside outside of this namespace.

```
class ActionMountGnomeIOLocation : public sh::actions::ActionActionItem
    #include <gnomeiofilesystemhandler.h> Action for mounting a GIO location.
```

## Public Functions

```
ActionMountGnomeIOLocation (QList<std::shared_ptr<sh::filesystem::FilesystemNode>>
    nodes)
```

Constructed only internally.

```
void execute ()
    Executes this action.
```

```
void execute (sh::actions::ActionExecutionInfo *info)
    Executes this action.
```

```
QKeySequence shortcuthint ()
    Returns the keyboard shortcut for triggering this action.
```

bool **shortcutHintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcutHint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<AbstractActionItem> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<AbstractActionItem> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

## Public Static Functions

void **offerMountGVolumeUuid** (std::shared\_ptr<sh::filesystem::FilesystemNode> node, QString gvolumeuuid)

void **offerMountLocation** (std::shared\_ptr<sh::filesystem::FilesystemNode> node, QString location)

void **unofferMount** (std::shared\_ptr<sh::filesystem::FilesystemNode> node)

void **doInitialize** ()

void **doShutdown** ()

QString **mount** (sh::actions::ActionExecutionInfo \*info, QString gvolumeuuid, QString location, std::shared\_ptr<filesystem::FilesystemNode> node)

**class ActionMountNetworkGnomeIOLocation** : public sh::actions::ActionActionItem  
#include <gnomeionetworkfilesystemhandler.h> Action for mounting a GIO network location.

## Public Functions

**ActionMountNetworkGnomeIOLocation** ()  
Constructed only internally.

void **execute** ()  
Executes this action.

void **execute** (sh::actions::ActionExecutionInfo \*info)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence shortcut, bool triggersOnCurrentDirectory = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<AbstractActionItem> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_\_setParentAction** (std::shared\_ptr<AbstractActionItem> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>

> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **changed** ()

Emits when some data changed.

**class ActionUnmountGnomeIOLocation : public *sh::actions::ActionActionItem***

*#include <gnomeiofilesystemhandler.h>* Action for unmounting a GIO location.



## Public Functions

**ActionUnmountGnomeIOLocation** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>>  
nodes)

Constructed only internally.

void **execute** ()

Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \*info)

Executes this action.

QKeySequence **shortcuthint** ()

Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()

Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence shortcut, bool triggersOnCurrentDirectory = false)

Sets the keyboard shortcut for triggering this action.

QString **text** ()

Returns the displayed text for this action.

QString **icon** ()

Returns the icon for this action.

bool **enabled** ()

Checks if this action is enabled.

bool **isChecked** ()

Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () const

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString text)

Sets the displayed text.

void **setIcon** (QString icon)

Sets the icon.

void **setEnabled** (bool enabled)

Sets if the item is enabled.

void **setChecked** (bool checked)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool visible)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

`std::weak_ptr<AbstractActionItem> parentAction ()`  
Returns the parent action, if it is added to a container.

`void _setParentAction (std::shared_ptr<AbstractActionItem> parent)`  
Sets the parent action. .

`void initializeAsync (std::function<void>  
> oninitialized = 0)` Asynchronously initializes the action.

`void initializeSync ()`  
Synchronously initializes the action.

`bool isInitialized ()`  
Checks if this action is initialized.

`bool isInitializing ()`  
Checks if this action is initializing.

## Signals

`void changed ()`  
Emits when some data changed.

## Public Static Functions

`void offerUnmountLocation (std::shared_ptr<sh::filesystem::FilesystemNode> node,  
QString location, bool staticmode = false)`

`void unofferUnmount (std::shared_ptr<sh::filesystem::FilesystemNode> node)`

`void doInitialize ()`

`void doShutdown ()`

**class ArchiveFilesystemHandler** : public *sh::filesystem::FilesystemHandler*  
*#include <archivefilesystemhandler.h>* A filesystem handler for archive files of some formats.

This implementation uses process calls to some external utility tools.

## Public Functions

**ArchiveFilesystemHandler** (*sh::filesystem::FilesystemModel* \**model*)

`void itemlist (sh::filesystem::Operation *op, std::shared_ptr<const  
sh::filesystem::Eurl> eurl, sh::filesystem::FilesystemNodeType type,  
sh::filesystem::FilesystemNodeListEditor *list)`

`sh::filesystem::FilesystemNodeType getType (sh::filesystem::Operation *op,  
std::shared_ptr<const sh::filesystem::Eurl>  
eurl)`

`qint64 getSize (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>  
eurl)`

`QDateTime getMtime (sh::filesystem::Operation *op, std::shared_ptr<const  
sh::filesystem::Eurl> eurl)`

`void setMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>  
eurl, QDateTime time)`

```

QString getMimeType (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl)

QString getLinkTarget (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl)

bool canGetFileContent (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl)

std::shared_ptr<QIODevice> getFileContent (sh::filesystem::Operation
                                         *op, std::shared_ptr<const
                                         sh::filesystem::Eurl> eurl)

bool canCreateDirectory (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl)

void createDirectory (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl)

bool canCreateLink (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl)

void createLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl, QString target)

bool canCreateFile (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl)

void createFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl, QIODevice *content, HandlerTransfer *handlertransfer)

bool canDeleteItem (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl)

void deleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl)

bool canRenameItem (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> src)

void renameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    src, QString destpath)

QList<std::shared_ptr<sh::actions::AbstractActionItem>> getActions (const
                                                                QList<std::shared_ptr<const
                                                                sh::filesystem::Eurl>>
                                                                eurls)

void itemlist (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl, sh::filesystem::FilesystemNodeType type,
                    sh::filesystem::FilesystemNodeListEditor *list) = 0
    Determine a list of subelements in a certain directory.

void configureItems (sh::filesystem::Operation *op, QList<std::shared_ptr<FilesystemNode>>
                    nodes)
    Configure newly created nodes (e.g. setting another icon or changing the display name).

sh::filesystem::FilesystemNodeType getType (sh::filesystem::Operation *op,
                                         std::shared_ptr<const sh::filesystem::Eurl>
                                         eurl) = 0
    Determine if a file is regular, or a dir, or a link, ...

qint64 getSize (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl) = 0
    Determine the size of a file.

```

`QDateTime getMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl) = 0`  
Determine the mtime of a file.

`void setMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl, QDateTime time) = 0`  
Set the mtime of a file.

`QString getMimeType (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl) = 0`  
Determine mime type of a file.

`QString getLinkTarget (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl) = 0`  
Resolve a link.

`QList<QString> listExtendedAttributes (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl)`  
Fetches the list of available extended attributes for an entry.

`quint64 getExtendedAttributeSize (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl, QString attribute)`  
Returns the size of value for one extended attribute for an entry.

`QByteArray getExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl, QString attribute)`  
Returns the value for one extended attribute for an entry.

`void setExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl, QString attribute, QByteArray value)`  
Sets the value for one extended attribute for an entry.

`void removeExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl, QString attribute)`  
Removes one extended attributes for an entry.

`QMap<QString, QString> getCustomAttributes (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl)`  
Returns the custom attributes for an entry.

In contrast to extended attributes, the custom attributes are not directly stored in the filesystem this way, but handle implementation specific aspects (like permissions).

`void setCustomAttribute (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl, QString attribute, QString value)`  
Sets the value for one custom attribute for an entry.  
See also `getCustomAttributes`.

`bool canGetFileContent (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl) = 0`  
Returns if it is allowed to get the content of a certain file.

`std::shared_ptr<QIODevice> getFileContent (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl) = 0`  
Get the content of a file.

```

bool canCreateDirectory (sh::filesystem::Operation *op, std::shared_ptr<const
                        sh::filesystem::Eurl> eurl) = 0
    Returns if it is allowed to create subdirectories in a certain directory.

void createDirectory (sh::filesystem::Operation *op, std::shared_ptr<const
                        sh::filesystem::Eurl> eurl) = 0
    Create a directory.

bool canCreateLink (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) = 0
    Returns if it is allowed to create a link in a certain directory.

void createLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                eurl, QString target) = 0
    Create a link.

bool canCreateFile (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) = 0
    Returns if it is allowed to create files in a certain directory.

void createFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                eurl, QIODevice *content, HandlerTransfer *handlertransfer) = 0
    Creates a file with some content.

    It may use handlertransfer for some better integration, like cancel support and progress
    monitoring.

bool canDeleteItem (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) = 0
    Returns if it is allowed to delete a certain file/directory/link/...

void deleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                eurl) = 0
    Delete a file/directory/link/...

void deleteDirectoryIfEmpty (sh::filesystem::Operation *op, std::shared_ptr<const
                            sh::filesystem::Eurl> eurl)
    Deletes a directory only if it is empty.

bool canRenameItem (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> src) = 0
    Returns if it is allowed to rename a certain file, directory, link, ... (see renameItem).

void renameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                src, QString destpath) = 0
    Renames an item to another path.

    It can be a simple filename change or also changes in the deeper path segments.

QList<std::shared_ptr<sh::actions::AbstractActionItem>> getActions (const
                                                                QList<std::shared_ptr<const
                                                                sh::filesystem::Eurl>>
                                                                eurls) = 0
    Returns a list of actions (see former example) for showing for certain files.

bool requiresResolvedLinks ()
    Returns if this handler requires to be used by the engine with resolved links.

void viewEnteredDirectory (std::shared_ptr<const sh::filesystem::Eurl>)
    Callback for preparing stuff when the view entered some directory (e.g. for enabling watchers).
    See also viewLeftDirectory.

void viewLeftDirectory (std::shared_ptr<const sh::filesystem::Eurl>)
    Callback for preparing stuff when the view left some directory. See also viewEnteredDirectory.

```

bool **isWellKnownScheme** ()

Returns if the scheme for this handler is a well known one (which external programs typically would understand).

void **search** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl*> eurl, std::function<void> std::shared\_ptr<const  
*sh::filesystem::Eurl*>, QList<std::shared\_ptr<*sh::search::SearchCriterion*>>,  
*sh::filesystem::FilesystemNodeType* ftype  
> addfile, std::function<void>std::shared\_ptr<const *sh::filesystem::Eurl*>> visitdir,  
QList<std::shared\_ptr<*sh::search::SearchCriterion*>> criteria) Helps searching for files.

Override this method with a behavior identical to the default, if a specialized implementation can be much faster than the default one.

void **customizeUi** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> node, bool \*showSearch-  
Panel)

Creates a custom gui, which becomes part of the fileview.

*sh::filesystem::FilesystemModel* \***model** ()

A convenience shortcut to the *sh::filesystem::FilesystemModel* (a singleton).

## Public Static Functions

void **doInitialize** ()

void **doShutdown** ()

**class GnomeIODevicesFilesystemHandler**: public *sh::filesystemhandlers::GnomeIOFilesystemHandler*, public  
#include <gnomeiodevicesfilesystemhandler.h> A filesystem handler for the GIO filesystem 'Devices'  
subtree.

## Public Functions

~**GnomeIODevicesFilesystemHandler** ()

void **itemlist** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl*> eurl, *sh::filesystem::FilesystemNodeType* type,  
*sh::filesystem::FilesystemNodeListEditor* \*list) **override**

*sh::filesystem::FilesystemNodeType* **getType** (*sh::filesystem::Operation* \*op,  
std::shared\_ptr<const *sh::filesystem::Eurl*>  
eurl) **override**

QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> **getActions** (const  
QList<std::shared\_ptr<const  
*sh::filesystem::Eurl*>>  
eurls) **override**

void **refresh** ()  
Refreshes the device list.

void **requestNewNode** (std::shared\_ptr<const *sh::filesystem::Eurl*> eurl,  
std::shared\_ptr<*sh::filesystem::FilesystemNode*> \*nnode)  
Requests a new device node.

void **itemlist** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl*> eurl, *sh::filesystem::FilesystemNodeType* type,  
*sh::filesystem::FilesystemNodeListEditor* \*list) = 0  
Determine a list of subelements in a certain directory.

```

sh::filesystem::FilesystemNodeType getType (sh::filesystem::Operation *op,
                                             std::shared_ptr<const sh::filesystem::Eurl>
                                             eurl) = 0
    Determine if a file is regular, or a dir, or a link, ...

qint64 getSize (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
               eurl)
qint64 getSize (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
               eurl) = 0
    Determine the size of a file.

QDateTime getMtime (sh::filesystem::Operation *op, std::shared_ptr<const
                  sh::filesystem::Eurl> eurl)
QDateTime getMtime (sh::filesystem::Operation *op, std::shared_ptr<const
                  sh::filesystem::Eurl> eurl) = 0
    Determine the mtime of a file.

void setMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
              eurl, QDateTime time)
void setMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
              eurl, QDateTime time) = 0
    Set the mtime of a file.

QString getMimeType (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl)
QString getMimeType (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) = 0
    Determine mime type of a file.

QString getLinkTarget (sh::filesystem::Operation *op, std::shared_ptr<const
                      sh::filesystem::Eurl> eurl)
QString getLinkTarget (sh::filesystem::Operation *op, std::shared_ptr<const
                      sh::filesystem::Eurl> eurl) = 0
    Resolve a link.

bool canGetFileContent (sh::filesystem::Operation *op, std::shared_ptr<const
                      sh::filesystem::Eurl> eurl)
bool canGetFileContent (sh::filesystem::Operation *op, std::shared_ptr<const
                      sh::filesystem::Eurl> eurl) = 0
    Returns if it is allowed to get the content of a certain file.

std::shared_ptr<QIODevice> getFileContent (sh::filesystem::Operation
                                           *op, std::shared_ptr<const
                                           sh::filesystem::Eurl> eurl)
std::shared_ptr<QIODevice> getFileContent (sh::filesystem::Operation
                                           *op, std::shared_ptr<const
                                           sh::filesystem::Eurl> eurl) = 0
    Get the content of a file.

bool canCreateDirectory (sh::filesystem::Operation *op, std::shared_ptr<const
                        sh::filesystem::Eurl> eurl)
bool canCreateDirectory (sh::filesystem::Operation *op, std::shared_ptr<const
                        sh::filesystem::Eurl> eurl) = 0
    Returns if it is allowed to create subdirectories in a certain directory.

void createDirectory (sh::filesystem::Operation *op, std::shared_ptr<const
                      sh::filesystem::Eurl> eurl)

```



```
void createDirectory (sh::filesystem::Operation *op,          std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) = 0
    Create a directory.

bool canCreateLink (sh::filesystem::Operation *op,          std::shared_ptr<const
                    sh::filesystem::Eurl> eurl)

bool canCreateLink (sh::filesystem::Operation *op,          std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) = 0
    Returns if it is allowed to create a link in a certain directory.

void createLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                eurl, QString target)

void createLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                eurl, QString target) = 0
    Create a link.

bool canCreateFile (sh::filesystem::Operation *op,          std::shared_ptr<const
                    sh::filesystem::Eurl> eurl)

bool canCreateFile (sh::filesystem::Operation *op,          std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) = 0
    Returns if it is allowed to create files in a certain directory.

void createFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                eurl, QIODevice *content, HandlerTransfer *handlertransfer)

void createFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                eurl, QIODevice *content, HandlerTransfer *handlertransfer) = 0
    Creates a file with some content.

    It may use handlertransfer for some better integration, like cancel support and progress
    monitoring.

bool canDeleteItem (sh::filesystem::Operation *op,          std::shared_ptr<const
                    sh::filesystem::Eurl> eurl)

bool canDeleteItem (sh::filesystem::Operation *op,          std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) = 0
    Returns if it is allowed to delete a certain file/directory/link/...

void deleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                eurl)

void deleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                eurl) = 0
    Delete a file/directory/link/...

bool canRenameItem (sh::filesystem::Operation *op,          std::shared_ptr<const
                    sh::filesystem::Eurl> src)

bool canRenameItem (sh::filesystem::Operation *op,          std::shared_ptr<const
                    sh::filesystem::Eurl> src) = 0
    Returns if it is allowed to rename a certain file, directory, link, ... (see renameItem).

void renameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                src, QString destpath)

void renameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                src, QString destpath) = 0
    Renames an item to another path.

    It can be a simple filename change or also changes in the deeper path segments.
```



`QList<std::shared_ptr<sh::actions::AbstractActionItem>> getActions (const  
QList<std::shared_ptr<const  
sh::filesystem::Eurl>>  
eurls) = 0`

Returns a list of actions (see former example) for showing for certain files.

`bool requiresResolvedLinks ()`

Returns if this handler requires to be used by the engine with resolved links.

`bool isWellKnownScheme ()`

Returns if the scheme for this handler is a well known one (which external programs typically would understand).

`void configureItems (sh::filesystem::Operation *op, QList<std::shared_ptr<FileSystemNode>>  
nodes)`

Configure newly created nodes (e.g. setting another icon or changing the display name).

`QList<QString> listExtendedAttributes (sh::filesystem::Operation *op,  
std::shared_ptr<const sh::filesystem::Eurl>  
eurl)`

Fetches the list of available extended attributes for an entry.

`quint64 getExtendedAttributeSize (sh::filesystem::Operation *op,  
std::shared_ptr<const sh::filesystem::Eurl>  
eurl, QString attribute)`

Returns the size of value for one extended attribute for an entry.

`QByteArray getExtendedAttribute (sh::filesystem::Operation *op,  
std::shared_ptr<const sh::filesystem::Eurl>  
eurl, QString attribute)`

Returns the value for one extended attribute for an entry.

`void setExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const  
sh::filesystem::Eurl> eurl, QString attribute, QByteArray  
value)`

Sets the value for one extended attribute for an entry.

`void removeExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const  
sh::filesystem::Eurl> eurl, QString attribute)`

Removes one extended attributes for an entry.

`QMap<QString, QString> getCustomAttributes (sh::filesystem::Operation  
*op, std::shared_ptr<const  
sh::filesystem::Eurl> eurl)`

Returns the custom attributes for an entry.

In contrast to extended attributes, the custom attributes are not directly stored in the filesystem this way, but handle implementation specific aspects (like permissions).

`void setCustomAttribute (sh::filesystem::Operation *op, std::shared_ptr<const  
sh::filesystem::Eurl> eurl, QString attribute, QString value)`

Sets the value for one custom attribute for an entry.

See also `getCustomAttributes`.

`void deleteDirectoryIfEmpty (sh::filesystem::Operation *op, std::shared_ptr<const  
sh::filesystem::Eurl> eurl)`

Deletes a directory only if it is empty.

`void viewEnteredDirectory (std::shared_ptr<const sh::filesystem::Eurl>)`

Callback for preparing stuff when the view entered some directory (e.g. for enabling watchers). See also `viewLeftDirectory`.

void **viewLeftDirectory** (std::shared\_ptr<const *sh::filesystem::Eurl*>)

Callback for preparing stuff when the view left some directory. See also `viewEnteredDirectory`.

void **search** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl*> eurl, std::function<void> std::shared\_ptr<const  
*sh::filesystem::Eurl*>, QList<std::shared\_ptr<*sh::search::SearchCriterion*>>,  
*sh::filesystem::FilesystemNodeType* ftype  
> addfile, std::function<void>std::shared\_ptr<const *sh::filesystem::Eurl*>> visitdir,  
QList<std::shared\_ptr<*sh::search::SearchCriterion*>> criteriaHelps searching for files.

Override this method with a behavior identical to the default, if a specialized implementation can be much faster than the default one.

void **customizeUi** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> node, bool \*showSearch-  
Panel)

Creates a custom gui, which becomes part of the fileview.

*sh::filesystem::FilesystemModel* \*model ()

A convenience shortcut to the *sh::filesystem::FilesystemModel* (a singleton).

void **doShutdown** ()

Executes singleton shutdown.

void **shutdown** ()

Shutdown down this singleton.

bool **isAlive** ()

Returns if this singleton is alive (true until its shutdown begins).

## Public Static Functions

QString **mountDevice** (QString gvolumeuuid, *sh::actions::ActionExecutionInfo* \*info)

QString **mountLocation** (QString gurl, *sh::actions::ActionExecutionInfo* \*info)

QString **unmountLocation** (QString gurl, *sh::actions::ActionExecutionInfo* \*info)

QByteArray **eurl2gurlb** (std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)

QString **eurl2gurl** (std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)

std::shared\_ptr<const *sh::filesystem::Eurl*> **gurl2eurl** (QString gurl)

void **logIfError** (void \*error)

void **throwIfError** (void \*error)

## Private Functions

**GnomeIODevicesFilesystemHandler** ()

## Friends

**friend class** ActionMount

**class** GnomeIOFilesystemHandler : public *sh::filesystem::FilesystemHandler*  
*#include <gnomeiofilesystemhandler.h>* A filesystem handler for GIO filesystem parts.

This is the abstract base class.

Subclassed by *sh::filesystemhandlers::GnomeIODevicesFilesystemHandler*,  
*sh::filesystemhandlers::GnomeIOGenericFilesystemHandler*, *sh::filesystemhandlers::GnomeIONetworkFilesystemHandler*,  
*sh::filesystemhandlers::GnomeIOSmbFilesystemHandler*

## Public Functions

**GnomeIOFilesystemHandler** ()

void **itemlist** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl>* eurl, *sh::filesystem::FilesystemNodeType* type,  
*sh::filesystem::FilesystemNodeListEditor* \*list)

*sh::filesystem::FilesystemNodeType* **getType** (*sh::filesystem::Operation* \*op,  
std::shared\_ptr<const *sh::filesystem::Eurl>*  
eurl)

qint64 **getSize** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl>*  
eurl)

QDateTime **getMtime** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl>* eurl)

void **setMtime** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl>*  
eurl, QDateTime time)

QString **getMimeType** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl>* eurl)

QString **getLinkTarget** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl>* eurl)

bool **canGetFileContent** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl>* eurl)

std::shared\_ptr<QIODevice> **getFileContent** (*sh::filesystem::Operation*  
\*op, std::shared\_ptr<const  
*sh::filesystem::Eurl>* eurl)

bool **canCreateDirectory** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl>* eurl)

void **createDirectory** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl>* eurl)

bool **canCreateLink** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl>* eurl)

void **createLink** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl>*  
eurl, QString target)

bool **canCreateFile** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl>* eurl)

void **createFile** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl>*  
eurl, QIODevice \*content, HandlerTransfer \*handlertransfer)

```
bool canDeleteItem (sh::filesystem::Operation *op, std::shared_ptr<const  
                  sh::filesystem::Eurl> eurl)  
void deleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>  
                eurl)  
bool canRenameItem (sh::filesystem::Operation *op, std::shared_ptr<const  
                  sh::filesystem::Eurl> src)  
void renameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>  
                src, QString destpath)  
QList<std::shared_ptr<sh::actions::AbstractActionItem>> getActions (const  
                                                                    QList<std::shared_ptr<const  
                                                                    sh::filesystem::Eurl>>  
                                                                    eurls)  
  
bool requiresResolvedLinks ()  
    Returns if this handler requires to be used by the engine with resolved links.  
  
bool isWellKnownScheme ()  
    Returns if the scheme for this handler is a well known one (which external programs typically  
    would understand).  
  
void itemlist (sh::filesystem::Operation *op, std::shared_ptr<const  
            sh::filesystem::Eurl> eurl, sh::filesystem::FileSystemNodeType type,  
            sh::filesystem::FileSystemNodeListEditor *list) = 0  
    Determine a list of subelements in a certain directory.  
  
void configureItems (sh::filesystem::Operation *op, QList<std::shared_ptr<FileSystemNode>>  
                    nodes)  
    Configure newly created nodes (e.g. setting another icon or changing the display name).  
  
sh::filesystem::FileSystemNodeType getType (sh::filesystem::Operation *op,  
                                            std::shared_ptr<const sh::filesystem::Eurl>  
                                            eurl) = 0  
    Determine if a file is regular, or a dir, or a link, ...  
  
qint64 getSize (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>  
                eurl) = 0  
    Determine the size of a file.  
  
QDateTime getMtime (sh::filesystem::Operation *op, std::shared_ptr<const  
                sh::filesystem::Eurl> eurl) = 0  
    Determine the mtime of a file.  
  
void setMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>  
                eurl, QDateTime time) = 0  
    Set the mtime of a file.  
  
QString getMimeType (sh::filesystem::Operation *op, std::shared_ptr<const  
                sh::filesystem::Eurl> eurl) = 0  
    Determine mime type of a file.  
  
QString getLinkTarget (sh::filesystem::Operation *op, std::shared_ptr<const  
                sh::filesystem::Eurl> eurl) = 0  
    Resolve a link.  
  
QList<QString> listExtendedAttributes (sh::filesystem::Operation *op,  
                                        std::shared_ptr<const sh::filesystem::Eurl>  
                                        eurl)  
    Fetches the list of available extended attributes for an entry.
```

```

quint64 getExtendedAttributeSize (sh::filesystem::Operation *op,      std::shared_ptr<const sh::filesystem::Eurl>
                                std::shared_ptr<const eurl, QString attribute>)
    Returns the site of value for one extended attribute for an entry.

QByteArray getExtendedAttribute (sh::filesystem::Operation *op,      std::shared_ptr<const sh::filesystem::Eurl>
                                std::shared_ptr<const eurl, QString attribute>)
    Returns the value for one extended attribute for an entry.

void setExtendedAttribute (sh::filesystem::Operation *op,      std::shared_ptr<const sh::filesystem::Eurl> eurl, QString attribute, QByteArray
                                value)
    Sets the value for one extended attribute for an entry.

void removeExtendedAttribute (sh::filesystem::Operation *op,      std::shared_ptr<const sh::filesystem::Eurl> eurl, QString attribute)
    Removes one extended attributes for an entry.

QMap<QString, QString> getCustomAttributes (sh::filesystem::Operation
                                             *op,      std::shared_ptr<const sh::filesystem::Eurl> eurl)
    Returns the custom attributes for an entry.

    In contrast to extended attributes, the custom attributes are not directly stored in the filesystem
    this way, but handle implementation specific aspects (like permissions).

void setCustomAttribute (sh::filesystem::Operation *op,      std::shared_ptr<const sh::filesystem::Eurl> eurl, QString attribute, QString value)
    Sets the value for one custom attribute for an entry.

    See also getCustomAttributes.

bool canGetFileContent (sh::filesystem::Operation *op,      std::shared_ptr<const sh::filesystem::Eurl> eurl) = 0
    Returns if it is allowed to get the content of a certain file.

std::shared_ptr<QIODevice> getFileContent (sh::filesystem::Operation
                                           *op,      std::shared_ptr<const sh::filesystem::Eurl> eurl) = 0
    Get the content of a file.

bool canCreateDirectory (sh::filesystem::Operation *op,      std::shared_ptr<const sh::filesystem::Eurl> eurl) = 0
    Returns if it is allowed to create subdirectories in a certain directory.

void createDirectory (sh::filesystem::Operation *op,      std::shared_ptr<const sh::filesystem::Eurl> eurl) = 0
    Create a directory.

bool canCreateLink (sh::filesystem::Operation *op,      std::shared_ptr<const sh::filesystem::Eurl> eurl) = 0
    Returns if it is allowed to create a link in a certain directory.

void createLink (sh::filesystem::Operation *op,      std::shared_ptr<const sh::filesystem::Eurl>
                eurl, QString target) = 0
    Create a link.

bool canCreateFile (sh::filesystem::Operation *op,      std::shared_ptr<const sh::filesystem::Eurl> eurl) = 0
    Returns if it is allowed to create files in a certain directory.

```

void **createFile** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*>  
                  *eurl*, QIODevice \*content, HandlerTransfer \*handlertransfer) = 0  
Creates a file with some content.

It may use handlertransfer for some better integration, like cancel support and progress monitoring.

bool **canDeleteItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
                      *sh::filesystem::Eurl*> *eurl*) = 0  
Returns if it is allowed to delete a certain file/directory/link/...

void **deleteItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*>  
                  *eurl*) = 0  
Delete a file/directory/link/...

void **deleteDirectoryIfEmpty** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
                              *sh::filesystem::Eurl*> *eurl*)  
Deletes a directory only if it is empty.

bool **canRenameItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
                      *sh::filesystem::Eurl*> *src*) = 0  
Returns if it is allowed to rename a certain file, directory, link, ... (see renameItem).

void **renameItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*>  
                  *src*, QString *destpath*) = 0  
Renames an item to another path.

It can be a simple filename change or also changes in the deeper path segments.

QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> **getActions** (const  
  QList<std::shared\_ptr<const  
  *sh::filesystem::Eurl*>>  
  *eurls*) = 0  
Returns a list of actions (see former example) for showing for certain files.

void **viewEnteredDirectory** (std::shared\_ptr<const *sh::filesystem::Eurl*>)  
Callback for preparing stuff when the view entered some directory (e.g. for enabling watchers).  
See also viewLeftDirectory.

void **viewLeftDirectory** (std::shared\_ptr<const *sh::filesystem::Eurl*>)  
Callback for preparing stuff when the view left some directory. See also viewEnteredDirectory.

void **search** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
                  *sh::filesystem::Eurl*> *eurl*, std::function<void> std::shared\_ptr<const  
                  *sh::filesystem::Eurl*>, QList<std::shared\_ptr<*sh::search::SearchCriterion*>>, *sh::filesystem::FilesystemNodeType* ftype  
> *addfile*, std::function<void> std::shared\_ptr<const *sh::filesystem::Eurl*>> *visitdir*,  
QList<std::shared\_ptr<*sh::search::SearchCriterion*>> *criteria*)  
Helps searching for files.

Override this method with a behavior identical to the default, if a specialized implementation can be much faster than the default one.

void **customizeUi** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> *node*, bool \*showSearch-  
                    *Panel*)  
Creates a custom gui, which becomes part of the fileview.

*sh::filesystem::FilesystemModel* \***model** ()  
A convenience shortcut to the *sh::filesystem::FilesystemModel* (a singleton).

## Public Static Functions

```

QString mountDevice (QString gvolumeuuid, sh::actions::ActionExecutionInfo *info)
QString mountLocation (QString gurl, sh::actions::ActionExecutionInfo *info)
QString unmountLocation (QString gurl, sh::actions::ActionExecutionInfo *info)
QByteArray eurl2gurlb (std::shared_ptr<const sh::filesystem::Eurl> eurl)
QString eurl2gurl (std::shared_ptr<const sh::filesystem::Eurl> eurl)
std::shared_ptr<const sh::filesystem::Eurl> gurl2eurl (QString gurl)
void logIfError (void *error)
void throwIfError (void *error)
class MountOperations
    #include <gnomeiofilesystemhandler.h>

```

## Public Static Functions

```

void *createMountOperation (sh::actions::ActionExecutionInfo *info, QString address,
                             MountOperationsCallbackData **opdata)
void deleteMountOperation (MountOperationsCallbackData *opdata)

```

## Public Static Attributes

```

QHash<size_t, MountOperationsCallbackData*> callbackdata
size_t callbackdataindex
QMutex callbackdatamutex
struct MountOperationsCallbackData
    #include <gnomeiofilesystemhandler.h>

```

## Public Functions

```

MountOperationsCallbackData (sh::actions::ActionExecutionInfo *info, size_t index,
                               void *mountoperation, QString address)

```

## Public Members

```

sh::actions::ActionExecutionInfo *info
size_t index
void *mountoperation
QString address
class GnomeIOGenericFilesystemHandler : public sh::filesystemhandlers::GnomeIOFilesystemHandler
    #include <gnomeiogenericfilesystemhandler.h> A filesystem handler for common/unspecial GIO
    filesystem parts.

```

This is a generic non-abstract implementation without special behavior.



## Public Functions

**GnomeIOGenericFilesystemHandler** (*sh::filesystem::FilesystemModel \*model*)

void **itemlist** (*sh::filesystem::Operation \*op,* *std::shared\_ptr<const sh::filesystem::Eurl> eurl,* *sh::filesystem::FilesystemNodeType type,* *sh::filesystem::FilesystemNodeListEditor \*list*)

void **itemlist** (*sh::filesystem::Operation \*op,* *std::shared\_ptr<const sh::filesystem::Eurl> eurl,* *sh::filesystem::FilesystemNodeType type,* *sh::filesystem::FilesystemNodeListEditor \*list*) = 0

Determine a list of subelements in a certain directory.

*sh::filesystem::FilesystemNodeType* **getType** (*sh::filesystem::Operation \*op,* *std::shared\_ptr<const sh::filesystem::Eurl> eurl*)

*sh::filesystem::FilesystemNodeType* **getType** (*sh::filesystem::Operation \*op,* *std::shared\_ptr<const sh::filesystem::Eurl> eurl*) = 0

Determine if a file is regular, or a dir, or a link, ...

qint64 **getSize** (*sh::filesystem::Operation \*op,* *std::shared\_ptr<const sh::filesystem::Eurl> eurl*)

qint64 **getSize** (*sh::filesystem::Operation \*op,* *std::shared\_ptr<const sh::filesystem::Eurl> eurl*) = 0

Determine the size of a file.

QDateTime **getMtime** (*sh::filesystem::Operation \*op,* *std::shared\_ptr<const sh::filesystem::Eurl> eurl*)

QDateTime **getMtime** (*sh::filesystem::Operation \*op,* *std::shared\_ptr<const sh::filesystem::Eurl> eurl*) = 0

Determine the mtime of a file.

void **setMtime** (*sh::filesystem::Operation \*op,* *std::shared\_ptr<const sh::filesystem::Eurl> eurl,* *QDateTime time*)

void **setMtime** (*sh::filesystem::Operation \*op,* *std::shared\_ptr<const sh::filesystem::Eurl> eurl,* *QDateTime time*) = 0

Set the mtime of a file.

QString **getMimetype** (*sh::filesystem::Operation \*op,* *std::shared\_ptr<const sh::filesystem::Eurl> eurl*)

QString **getMimetype** (*sh::filesystem::Operation \*op,* *std::shared\_ptr<const sh::filesystem::Eurl> eurl*) = 0

Determine mime type of a file.

QString **getLinkTarget** (*sh::filesystem::Operation \*op,* *std::shared\_ptr<const sh::filesystem::Eurl> eurl*)

QString **getLinkTarget** (*sh::filesystem::Operation \*op,* *std::shared\_ptr<const sh::filesystem::Eurl> eurl*) = 0

Resolve a link.

bool **canGetFileContent** (*sh::filesystem::Operation \*op,* *std::shared\_ptr<const sh::filesystem::Eurl> eurl*)

bool **canGetFileContent** (*sh::filesystem::Operation \*op,* *std::shared\_ptr<const sh::filesystem::Eurl> eurl*) = 0

Returns if it is allowed to get the content of a certain file.



```
std::shared_ptr<QIODevice> getFileContent (sh::filesystem::Operation
                                           *op,          std::shared_ptr<const
                                           sh::filesystem::Eurl> eurl)
```

```
std::shared_ptr<QIODevice> getFileContent (sh::filesystem::Operation
                                           *op,          std::shared_ptr<const
                                           sh::filesystem::Eurl> eurl) = 0
```

Get the content of a file.

```
bool canCreateDirectory (sh::filesystem::Operation *op,          std::shared_ptr<const
                        sh::filesystem::Eurl> eurl)
```

```
bool canCreateDirectory (sh::filesystem::Operation *op,          std::shared_ptr<const
                        sh::filesystem::Eurl> eurl) = 0
```

Returns if it is allowed to create subdirectories in a certain directory.

```
void createDirectory (sh::filesystem::Operation *op,          std::shared_ptr<const
                     sh::filesystem::Eurl> eurl)
```

```
void createDirectory (sh::filesystem::Operation *op,          std::shared_ptr<const
                     sh::filesystem::Eurl> eurl) = 0
```

Create a directory.

```
bool canCreateLink (sh::filesystem::Operation *op,          std::shared_ptr<const
                   sh::filesystem::Eurl> eurl)
```

```
bool canCreateLink (sh::filesystem::Operation *op,          std::shared_ptr<const
                   sh::filesystem::Eurl> eurl) = 0
```

Returns if it is allowed to create a link in a certain directory.

```
void createLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                eurl, QString target)
```

```
void createLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                eurl, QString target) = 0
```

Create a link.

```
bool canCreateFile (sh::filesystem::Operation *op,          std::shared_ptr<const
                   sh::filesystem::Eurl> eurl)
```

```
bool canCreateFile (sh::filesystem::Operation *op,          std::shared_ptr<const
                   sh::filesystem::Eurl> eurl) = 0
```

Returns if it is allowed to create files in a certain directory.

```
void createFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                eurl, QIODevice *content, HandlerTransfer *handlertransfer)
```

```
void createFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                eurl, QIODevice *content, HandlerTransfer *handlertransfer) = 0
```

Creates a file with some content.

It may use handlertransfer for some better integration, like cancel support and progress monitoring.

```
bool canDeleteItem (sh::filesystem::Operation *op,          std::shared_ptr<const
                   sh::filesystem::Eurl> eurl)
```

```
bool canDeleteItem (sh::filesystem::Operation *op,          std::shared_ptr<const
                   sh::filesystem::Eurl> eurl) = 0
```

Returns if it is allowed to delete a certain file/directory/link/...

```
void deleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                 eurl)
```

```
void deleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                eurl) = 0
    Delete a file/directory/link/...
```

```
bool canRenameItem (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> src)

bool canRenameItem (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> src) = 0
    Returns if it is allowed to rename a certain file, directory, link, ... (see renameItem).
```

```
void renameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                 src, QString destpath)

void renameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                 src, QString destpath) = 0
    Renames an item to another path.

    It can be a simple filename change or also changes in the deeper path segments.
```

```
QList<std::shared_ptr<sh::actions::AbstractActionItem>> getActions (const
                                                                QList<std::shared_ptr<const
                                                                sh::filesystem::Eurl>>
                                                                eurls)

QList<std::shared_ptr<sh::actions::AbstractActionItem>> getActions (const
                                                                QList<std::shared_ptr<const
                                                                sh::filesystem::Eurl>>
                                                                eurls) = 0

    Returns a list of actions (see former example) for showing for certain files.
```

```
bool requiresResolvedLinks ()
    Returns if this handler requires to be used by the engine with resolved links.
```

```
bool isWellKnownScheme ()
    Returns if the scheme for this handler is a well known one (which external programs typically
    would understand).
```

```
void configureItems (sh::filesystem::Operation *op, QList<std::shared_ptr<FilesystemNode>>
                    nodes)
    Configure newly created nodes (e.g. setting another icon or changing the display name).
```

```
QList<QString> listExtendedAttributes (sh::filesystem::Operation *op,
                                        std::shared_ptr<const sh::filesystem::Eurl>
                                        eurl)
    Fetches the list of available extended attributes for an entry.
```

```
quint64 getExtendedAttributeSize (sh::filesystem::Operation *op,
                                   std::shared_ptr<const sh::filesystem::Eurl>
                                   eurl, QString attribute)
    Returns the size of value for one extended attribute for an entry.
```

```
QByteArray getExtendedAttribute (sh::filesystem::Operation *op,
                                  std::shared_ptr<const sh::filesystem::Eurl>
                                  eurl, QString attribute)
    Returns the value for one extended attribute for an entry.
```

```
void setExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
                           sh::filesystem::Eurl> eurl, QString attribute, QByteArray
                           value)
    Sets the value for one extended attribute for an entry.
```

void **removeExtendedAttribute** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QString attribute)

Removes one extended attributes for an entry.

QMap<QString, QString> **getCustomAttributes** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)

Returns the custom attributes for an entry.

In contrast to extended attributes, the custom attributes are not directly stored in the filesystem this way, but handle implementation specific aspects (like permissions).

void **setCustomAttribute** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QString attribute, QString value)

Sets the value for one custom attribute for an entry.

See also getCustomAttributes.

void **deleteDirectoryIfEmpty** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)

Deletes a directory only if it is empty.

void **viewEnteredDirectory** (std::shared\_ptr<const *sh::filesystem::Eurl*>)

Callback for preparing stuff when the view entered some directory (e.g. for enabling watchers). See also viewLeftDirectory.

void **viewLeftDirectory** (std::shared\_ptr<const *sh::filesystem::Eurl*>)

Callback for preparing stuff when the view left some directory. See also viewEnteredDirectory.

void **search** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, std::function<void> std::shared\_ptr<const *sh::filesystem::Eurl*>, QList<std::shared\_ptr<*sh::search::SearchCriterion*>>, *sh::filesystem::FilesystemNodeType* ftype > addfile, std::function<void>std::shared\_ptr<const *sh::filesystem::Eurl*>> visitdir, QList<std::shared\_ptr<*sh::search::SearchCriterion*>> criteriaHelps searching for files.

Override this method with a behavior identical to the default, if a specialized implementation can be much faster than the default one.

void **customizeUi** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> node, bool \*showSearch-Panel)

Creates a custom gui, which becomes part of the fileview.

*sh::filesystem::FilesystemModel* \***model** ()

A convenience shortcut to the *sh::filesystem::FilesystemModel* (a singleton).

## Public Static Functions

QString **mountDevice** (QString gvolumeuuid, *sh::actions::ActionExecutionInfo* \*info)

QString **mountLocation** (QString gurl, *sh::actions::ActionExecutionInfo* \*info)

QString **unmountLocation** (QString gurl, *sh::actions::ActionExecutionInfo* \*info)

QByteArray **eurl2gurlb** (std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)

QString **eurl2gurl** (std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)

std::shared\_ptr<const *sh::filesystem::Eurl*> **gurl2eurl** (QString gurl)

void **logIfError** (void \*error)

void **throwIfError** (void \*error)

```
class GnomeIONetworkFilesystemHandler : public sh::filesystemhandlers::GnomeIOFilesystemHandler, public
#include <gnomeionetworkfilesystemhandler.h> A filesystem handler for the GIO filesystem 'Net-
work' subtree.
```

## Public Functions

```
QList<std::shared_ptr<sh::actions::AbstractActionItem>> getActions (const
                                                                    QList<std::shared_ptr<const
                                                                    sh::filesystem::Eurl>>
                                                                    eurls) override

void requestNewNode (std::shared_ptr<const sh::filesystem::Eurl> eurl,
                    std::shared_ptr<sh::filesystem::FilesystemNode> *nnode)
    Requests a new network node.

void itemList (sh::filesystem::Operation *op, std::shared_ptr<const
            sh::filesystem::Eurl> eurl, sh::filesystem::FilesystemNodeType type,
            sh::filesystem::FilesystemNodeListEditor *list)

void itemList (sh::filesystem::Operation *op, std::shared_ptr<const
            sh::filesystem::Eurl> eurl, sh::filesystem::FilesystemNodeType type,
            sh::filesystem::FilesystemNodeListEditor *list) = 0
    Determine a list of subelements in a certain directory.

sh::filesystem::FilesystemNodeType getType (sh::filesystem::Operation *op,
                                           std::shared_ptr<const sh::filesystem::Eurl>
                                           eurl)

sh::filesystem::FilesystemNodeType getType (sh::filesystem::Operation *op,
                                           std::shared_ptr<const sh::filesystem::Eurl>
                                           eurl) = 0
    Determine if a file is regular, or a dir, or a link, ...

qint64 getSize (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
               eurl)

qint64 getSize (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
               eurl) = 0
    Determine the size of a file.

QDateTime getMtime (sh::filesystem::Operation *op, std::shared_ptr<const
                   sh::filesystem::Eurl> eurl)

QDateTime getMtime (sh::filesystem::Operation *op, std::shared_ptr<const
                   sh::filesystem::Eurl> eurl) = 0
    Determine the mtime of a file.

void setMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
              eurl, QDateTime time)

void setMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
              eurl, QDateTime time) = 0
    Set the mtime of a file.

QString getMimeType (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl)

QString getMimeType (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) = 0
    Determine mime type of a file.

QString getLinkTarget (sh::filesystem::Operation *op, std::shared_ptr<const
                      sh::filesystem::Eurl> eurl)
```

QString **getLinkTarget** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl> eurl*) = 0

Resolve a link.

bool **canGetFileContent** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl> eurl*)

bool **canGetFileContent** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl> eurl*) = 0

Returns if it is allowed to get the content of a certain file.

std::shared\_ptr<QIODevice> **getFileContent** (*sh::filesystem::Operation*  
\*op, std::shared\_ptr<const  
*sh::filesystem::Eurl> eurl*)

std::shared\_ptr<QIODevice> **getFileContent** (*sh::filesystem::Operation*  
\*op, std::shared\_ptr<const  
*sh::filesystem::Eurl> eurl*) = 0

Get the content of a file.

bool **canCreateDirectory** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl> eurl*)

bool **canCreateDirectory** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl> eurl*) = 0

Returns if it is allowed to create subdirectories in a certain directory.

void **createDirectory** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl> eurl*)

void **createDirectory** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl> eurl*) = 0

Create a directory.

bool **canCreateLink** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl> eurl*)

bool **canCreateLink** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl> eurl*) = 0

Returns if it is allowed to create a link in a certain directory.

void **createLink** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl>*  
*eurl*, QString *target*)

void **createLink** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl>*  
*eurl*, QString *target*) = 0

Create a link.

bool **canCreateFile** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl> eurl*)

bool **canCreateFile** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl> eurl*) = 0

Returns if it is allowed to create files in a certain directory.

void **createFile** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl>*  
*eurl*, QIODevice \**content*, HandlerTransfer \**handlertransfer*)

void **createFile** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl>*  
*eurl*, QIODevice \**content*, HandlerTransfer \**handlertransfer*) = 0

Creates a file with some content.

It may use *handlertransfer* for some better integration, like cancel support and progress monitoring.

```
bool canDeleteItem (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl)
bool canDeleteItem (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) = 0
    Returns if it is allowed to delete a certain file/directory/link/...
```

```
void deleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                eurl)
void deleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                eurl) = 0
    Delete a file/directory/link/...
```

```
bool canRenameItem (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> src)
bool canRenameItem (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> src) = 0
    Returns if it is allowed to rename a certain file, directory, link, ... (see renameItem).
```

```
void renameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                 src, QString destpath)
void renameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                 src, QString destpath) = 0
    Renames an item to another path.

    It can be a simple filename change or also changes in the deeper path segments.
```

```
QList<std::shared_ptr<sh::actions::AbstractActionItem>> getActions (const
                                                                    QList<std::shared_ptr<const
                                                                    sh::filesystem::Eurl>>
                                                                    eurls) = 0
    Returns a list of actions (see former example) for showing for certain files.
```

```
bool requiresResolvedLinks ()
    Returns if this handler requires to be used by the engine with resolved links.
```

```
bool isWellKnownScheme ()
    Returns if the scheme for this handler is a well known one (which external programs typically
    would understand).
```

```
void configureItems (sh::filesystem::Operation *op, QList<std::shared_ptr<FilesystemNode>>
                     nodes)
    Configure newly created nodes (e.g. setting another icon or changing the display name).
```

```
QList<QString> listExtendedAttributes (sh::filesystem::Operation *op,
                                       std::shared_ptr<const sh::filesystem::Eurl>
                                       eurl)
    Fetches the list of available extended attributes for an entry.
```

```
quint64 getExtendedAttributeSize (sh::filesystem::Operation *op,
                                   std::shared_ptr<const sh::filesystem::Eurl>
                                   eurl, QString attribute)
    Returns the size of value for one extended attribute for an entry.
```

```
QByteArray getExtendedAttribute (sh::filesystem::Operation *op,
                                   std::shared_ptr<const sh::filesystem::Eurl>
                                   eurl, QString attribute)
    Returns the value for one extended attribute for an entry.
```



```
void setExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
                        sh::filesystem::Eurl> eurl, QString attribute, QByteArray
                        value)
    Sets the value for one extended attribute for an entry.

void removeExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
                        sh::filesystem::Eurl> eurl, QString attribute)
    Removes one extended attributes for an entry.

QMap<QString, QString> getCustomAttributes (sh::filesystem::Operation
                        *op, std::shared_ptr<const
                        sh::filesystem::Eurl> eurl)
    Returns the custom attributes for an entry.

    In contrast to extended attributes, the custom attributes are not directly stored in the filesystem
    this way, but handle implementation specific aspects (like permissions).

void setCustomAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
                        sh::filesystem::Eurl> eurl, QString attribute, QString value)
    Sets the value for one custom attribute for an entry.

    See also getCustomAttributes.

void deleteDirectoryIfEmpty (sh::filesystem::Operation *op, std::shared_ptr<const
                        sh::filesystem::Eurl> eurl)
    Deletes a directory only if it is empty.

void viewEnteredDirectory (std::shared_ptr<const sh::filesystem::Eurl>)
    Callback for preparing stuff when the view entered some directory (e.g. for enabling watchers).
    See also viewLeftDirectory.

void viewLeftDirectory (std::shared_ptr<const sh::filesystem::Eurl>)
    Callback for preparing stuff when the view left some directory. See also viewEnteredDirectory.

void search (sh::filesystem::Operation *op, std::shared_ptr<const
            sh::filesystem::Eurl> eurl, std::function<void> std::shared_ptr<const
            sh::filesystem::Eurl>, QList<std::shared_ptr<sh::search::SearchCriterion>>,
            sh::filesystem::FilesystemNodeType ftype
            > addfile, std::function<void>std::shared_ptr<const sh::filesystem::Eurl>> visitdir,
            QList<std::shared_ptr<sh::search::SearchCriterion>> criteria)
    Helps searching for files.

    Override this method with a behavior identical to the default, if a specialized implementation can
    be much faster than the default one.

void customizeUi (std::shared_ptr<sh::filesystem::FilesystemNode> node, bool *showSearch-
                Panel)
    Creates a custom gui, which becomes part of the fileview.

sh::filesystem::FilesystemModel *model ()
    A convenience shortcut to the sh::filesystem::FilesystemModel (a singleton).

void doShutdown ()
    Executes singleton shutdown.

void shutdown ()
    Shutdown down this singleton.

bool isAlive ()
    Returns if this singleton is alive (true until its shutdown begins).
```

## Public Static Functions

```
QString mountDevice (QString gvolumeuuid, sh::actions::ActionExecutionInfo *info)
QString mountLocation (QString gurl, sh::actions::ActionExecutionInfo *info)
QString unmountLocation (QString gurl, sh::actions::ActionExecutionInfo *info)
QByteArray eurl2gurlb (std::shared_ptr<const sh::filesystem::Eurl> eurl)
QString eurl2gurl (std::shared_ptr<const sh::filesystem::Eurl> eurl)
std::shared_ptr<const sh::filesystem::Eurl> gurl2eurl (QString gurl)
void logIfError (void *error)
void throwIfError (void *error)
```

## Private Functions

```
GnomeIONetworkFilesystemHandler ()
```

```
class GnomeIOSmbFilesystemHandler : public sh::filesystemhandlers::GnomeIOFilesystemHandler, public
#include <gnomeiosmbfilesystemhandler.h> A filesystem handler for the GIO filesystem ‘smb’ sub-
trees (in ‘Network’).
```

## Public Functions

```
sh::filesystem::FilesystemNodeType getType (sh::filesystem::Operation *op,
std::shared_ptr<const sh::filesystem::Eurl> eurl) override

void itemlist (sh::filesystem::Operation *op, std::shared_ptr<const
sh::filesystem::Eurl> eurl, sh::filesystem::FilesystemNodeType type,
sh::filesystem::FilesystemNodeListEditor *list)

void itemlist (sh::filesystem::Operation *op, std::shared_ptr<const
sh::filesystem::Eurl> eurl, sh::filesystem::FilesystemNodeType type,
sh::filesystem::FilesystemNodeListEditor *list) = 0
Determine a list of subelements in a certain directory.

sh::filesystem::FilesystemNodeType getType (sh::filesystem::Operation *op,
std::shared_ptr<const sh::filesystem::Eurl> eurl) = 0
Determine if a file is regular, or a dir, or a link, ...

qint64 getSize (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
eurl)

qint64 getSize (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
eurl) = 0
Determine the size of a file.

QDateTime getMtime (sh::filesystem::Operation *op, std::shared_ptr<const
sh::filesystem::Eurl> eurl)

QDateTime getMtime (sh::filesystem::Operation *op, std::shared_ptr<const
sh::filesystem::Eurl> eurl) = 0
Determine the mtime of a file.

void setMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
eurl, QDateTime time)
```



```

void setMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
               eurl, QDateTime time) = 0
    Set the mtime of a file.

QString getMimeType (sh::filesystem::Operation *op, std::shared_ptr<const
                   sh::filesystem::Eurl> eurl)

QString getMimeType (sh::filesystem::Operation *op, std::shared_ptr<const
                   sh::filesystem::Eurl> eurl) = 0
    Determine mime type of a file.

QString getLinkTarget (sh::filesystem::Operation *op, std::shared_ptr<const
                   sh::filesystem::Eurl> eurl)

QString getLinkTarget (sh::filesystem::Operation *op, std::shared_ptr<const
                   sh::filesystem::Eurl> eurl) = 0
    Resolve a link.

bool canGetFileContent (sh::filesystem::Operation *op, std::shared_ptr<const
                   sh::filesystem::Eurl> eurl)

bool canGetFileContent (sh::filesystem::Operation *op, std::shared_ptr<const
                   sh::filesystem::Eurl> eurl) = 0
    Returns if it is allowed to get the content of a certain file.

std::shared_ptr<QIODevice> getFileContent (sh::filesystem::Operation
                                           *op, std::shared_ptr<const
                                           sh::filesystem::Eurl> eurl)

std::shared_ptr<QIODevice> getFileContent (sh::filesystem::Operation
                                           *op, std::shared_ptr<const
                                           sh::filesystem::Eurl> eurl) = 0
    Get the content of a file.

bool canCreateDirectory (sh::filesystem::Operation *op, std::shared_ptr<const
                   sh::filesystem::Eurl> eurl)

bool canCreateDirectory (sh::filesystem::Operation *op, std::shared_ptr<const
                   sh::filesystem::Eurl> eurl) = 0
    Returns if it is allowed to create subdirectories in a certain directory.

void createDirectory (sh::filesystem::Operation *op, std::shared_ptr<const
                   sh::filesystem::Eurl> eurl)

void createDirectory (sh::filesystem::Operation *op, std::shared_ptr<const
                   sh::filesystem::Eurl> eurl) = 0
    Create a directory.

bool canCreateLink (sh::filesystem::Operation *op, std::shared_ptr<const
                   sh::filesystem::Eurl> eurl)

bool canCreateLink (sh::filesystem::Operation *op, std::shared_ptr<const
                   sh::filesystem::Eurl> eurl) = 0
    Returns if it is allowed to create a link in a certain directory.

void createLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                 eurl, QString target)

void createLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                 eurl, QString target) = 0
    Create a link.

bool canCreateFile (sh::filesystem::Operation *op, std::shared_ptr<const
                   sh::filesystem::Eurl> eurl)

```

bool **canCreateFile** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0  
Returns if it is allowed to create files in a certain directory.

void **createFile** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QIODevice \*content, HandlerTransfer \*handlertransfer)

void **createFile** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QIODevice \*content, HandlerTransfer \*handlertransfer) = 0  
Creates a file with some content.

It may use handlertransfer for some better integration, like cancel support and progress monitoring.

bool **canDeleteItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)

bool **canDeleteItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0  
Returns if it is allowed to delete a certain file/directory/link/...

void **deleteItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)

void **deleteItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0  
Delete a file/directory/link/...

bool **canRenameItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> src)

bool **canRenameItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> src) = 0  
Returns if it is allowed to rename a certain file, directory, link, ... (see renameItem).

void **renameItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> src, QString destpath)

void **renameItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> src, QString destpath) = 0  
Renames an item to another path.

It can be a simple filename change or also changes in the deeper path segments.

QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> **getActions** (const QList<std::shared\_ptr<const *sh::filesystem::Eurl*>> eurls)

QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> **getActions** (const QList<std::shared\_ptr<const *sh::filesystem::Eurl*>> eurls) = 0

Returns a list of actions (see former example) for showing for certain files.

bool **requiresResolvedLinks** ()

Returns if this handler requires to be used by the engine with resolved links.

bool **isWellKnownScheme** ()

Returns if the scheme for this handler is a well known one (which external programs typically would understand).

void **configureItems** (*sh::filesystem::Operation* \*op, QList<std::shared\_ptr<FilesystemNode>> nodes)

Configure newly created nodes (e.g. setting another icon or changing the display name).

`QList<QString> listExtendedAttributes (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl)`  
 Fetches the list of available extended attributes for an entry.

`quint64 getExtendedAttributeSize (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl, QString attribute)`  
 Returns the size of value for one extended attribute for an entry.

`QByteArray getExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl, QString attribute)`  
 Returns the value for one extended attribute for an entry.

`void setExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl, QString attribute, QByteArray value)`  
 Sets the value for one extended attribute for an entry.

`void removeExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl, QString attribute)`  
 Removes one extended attributes for an entry.

`QMap<QString, QString> getCustomAttributes (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl)`  
 Returns the custom attributes for an entry.

In contrast to extended attributes, the custom attributes are not directly stored in the filesystem this way, but handle implementation specific aspects (like permissions).

`void setCustomAttribute (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl, QString attribute, QString value)`  
 Sets the value for one custom attribute for an entry.

See also `getCustomAttributes`.

`void deleteDirectoryIfEmpty (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl)`  
 Deletes a directory only if it is empty.

`void viewEnteredDirectory (std::shared_ptr<const sh::filesystem::Eurl>)`  
 Callback for preparing stuff when the view entered some directory (e.g. for enabling watchers). See also `viewLeftDirectory`.

`void viewLeftDirectory (std::shared_ptr<const sh::filesystem::Eurl>)`  
 Callback for preparing stuff when the view left some directory. See also `viewEnteredDirectory`.

`void search (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl, std::function<void> std::shared_ptr<const sh::filesystem::Eurl>, QList<std::shared_ptr<sh::search::SearchCriterion>>, sh::filesystem::FilesystemNodeType ftype > addfile, std::function<void> std::shared_ptr<const sh::filesystem::Eurl>> visitdir, QList<std::shared_ptr<sh::search::SearchCriterion>> criteria)`  
 Helps searching for files.

Override this method with a behavior identical to the default, if a specialized implementation can be much faster than the default one.

`void customizeUi (std::shared_ptr<sh::filesystem::FilesystemNode> node, bool *showSearchPanel)`  
 Creates a custom gui, which becomes part of the fileview.

*sh::filesystem::FilesystemModel* \*model ()

A convenience shortcut to the *sh::filesystem::FilesystemModel* (a singleton).

void **doShutdown** ()

Executes singleton shutdown.

void **shutdown** ()

Shutdown down this singleton.

bool **isAlive** ()

Returns if this singleton is alive (true until its shutdown begins).

## Public Static Functions

QString **mountDevice** (QString gvolumeuuid, *sh::actions::ActionExecutionInfo* \*info)

QString **mountLocation** (QString gurl, *sh::actions::ActionExecutionInfo* \*info)

QString **unmountLocation** (QString gurl, *sh::actions::ActionExecutionInfo* \*info)

QByteArray **eurl2gurlb** (std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)

QString **eurl2gurl** (std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)

std::shared\_ptr<const *sh::filesystem::Eurl*> **gurl2eurl** (QString gurl)

void **logIfError** (void \*error)

void **throwIfError** (void \*error)

## Private Functions

**GnomeIOSmbFilesystemHandler** ()

**class LocalFilesystemHandler**: public *sh::filesystem::FilesystemHandler*, public *sh::base::Singleton*  
#include <localfilesystemhandler.h> A filesystem handler for the local filesystem.

Subclassed by *sh::filesystemhandlers::WindowsNetworkFilesystemHandler*

## Public Functions

void **itemlist** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, *sh::filesystem::FilesystemNodeType* type, *sh::filesystem::FilesystemNodeListEditor* \*list) **override**

void **configureItems** (*sh::filesystem::Operation* \*op, QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> nodes) **override**  
Configure newly created nodes (e.g. setting another icon or changing the display name).

*sh::filesystem::FilesystemNodeType* **getType** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) **override**

qint64 **getSize** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) **override**

QDateTime **getMtime** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) **override**

void **setMtime** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QDateTime mtime) **override**

```

QString getMimeType (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) override

QString getLinkTarget (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) override

bool canGetFileContent (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) override

std::shared_ptr<QIODevice> getFileContent (sh::filesystem::Operation
                    *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) override

bool canCreateDirectory (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) override

void createDirectory (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) override

bool canCreateLink (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) override

void createLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl, QString target) override

bool canCreateFile (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) override

void createFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl, QIODevice *content, HandlerTransfer *handlertransfer) override

bool canDeleteItem (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) override

void deleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl) override

bool canRenameItem (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> src) override

void renameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    src, QString destpath) override

QList<std::shared_ptr<sh::actions::AbstractActionItem>> getActions (const
                                                                    QList<std::shared_ptr<const
                                                                    sh::filesystem::Eurl>>
                                                                    eurls) override

QList<QString> listExtendedAttributes (sh::filesystem::Operation *op,
                    std::shared_ptr<const sh::filesystem::Eurl>
                    eurl) override

quint64 getExtendedAttributeSize (sh::filesystem::Operation *op,
                    std::shared_ptr<const sh::filesystem::Eurl>
                    eurl, QString attribute) override

QByteArray getExtendedAttribute (sh::filesystem::Operation *op,
                    std::shared_ptr<const sh::filesystem::Eurl>
                    eurl, QString attribute) override

void setExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl, QString attribute, QByteArray
                    value) override

void removeExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl, QString attribute)
override

```

```
QMap<QString, QString> getCustomAttributes (sh::filesystem::Operation
                                             *op,          std::shared_ptr<const
                                             sh::filesystem::Eurl> eurl) override

void setCustomAttribute (sh::filesystem::Operation *op,          std::shared_ptr<const
                        sh::filesystem::Eurl> eurl, QString attribute, QString value)
                        override

bool requiresResolvedLinks () override
    Returns if this handler requires to be used by the engine with resolved links.

bool isWellKnownScheme () override
    Returns if the scheme for this handler is a well known one (which external programs typically
    would understand).

QString eurlToLocal (std::shared_ptr<const sh::filesystem::Eurl> eurl)

std::shared_ptr<const sh::filesystem::Eurl> localToEurl (const QString local)

void itemlist (sh::filesystem::Operation *op,          std::shared_ptr<const
              sh::filesystem::Eurl> eurl, sh::filesystem::FileSystemNodeType type,
              sh::filesystem::FileSystemNodeListEditor *list) = 0
    Determine a list of subelements in a certain directory.

sh::filesystem::FileSystemNodeType getType (sh::filesystem::Operation *op,
                                             std::shared_ptr<const sh::filesystem::Eurl>
                                             eurl) = 0
    Determine if a file is regular, or a dir, or a link, ...

qint64 getSize (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
               eurl) = 0
    Determine the size of a file.

QDateTime getMtime (sh::filesystem::Operation *op,          std::shared_ptr<const
                   sh::filesystem::Eurl> eurl) = 0
    Determine the mtime of a file.

void setMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
              eurl, QDateTime time) = 0
    Set the mtime of a file.

QString getMimetype (sh::filesystem::Operation *op,          std::shared_ptr<const
                   sh::filesystem::Eurl> eurl) = 0
    Determine mime type of a file.

QString getLinkTarget (sh::filesystem::Operation *op,          std::shared_ptr<const
                      sh::filesystem::Eurl> eurl) = 0
    Resolve a link.

QList<QString> listExtendedAttributes (sh::filesystem::Operation *op,
                                       std::shared_ptr<const sh::filesystem::Eurl>
                                       eurl)
    Fetches the list of available extended attributes for an entry.

quint64 getExtendedAttributeSize (sh::filesystem::Operation *op,
                                   std::shared_ptr<const sh::filesystem::Eurl>
                                   eurl, QString attribute)
    Returns the size of value for one extended attribute for an entry.

QByteArray getExtendedAttribute (sh::filesystem::Operation *op,
                                  std::shared_ptr<const sh::filesystem::Eurl>
                                  eurl, QString attribute)
    Returns the value for one extended attribute for an entry.
```



```
void setExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
                        sh::filesystem::Eurl> eurl, QString attribute, QByteArray
                        value)
```

Sets the value for one extended attribute for an entry.

```
void removeExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
                        sh::filesystem::Eurl> eurl, QString attribute)
```

Removes one extended attributes for an entry.

```
QMap<QString, QString> getCustomAttributes (sh::filesystem::Operation
                        *op, std::shared_ptr<const
                        sh::filesystem::Eurl> eurl)
```

Returns the custom attributes for an entry.

In contrast to extended attributes, the custom attributes are not directly stored in the filesystem this way, but handle implementation specific aspects (like permissions).

```
void setCustomAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
                        sh::filesystem::Eurl> eurl, QString attribute, QString value)
```

Sets the value for one custom attribute for an entry.

See also getCustomAttributes.

```
bool canGetFileContent (sh::filesystem::Operation *op, std::shared_ptr<const
                        sh::filesystem::Eurl> eurl) = 0
```

Returns if it is allowed to get the content of a certain file.

```
std::shared_ptr<QIODevice> getFileContent (sh::filesystem::Operation
                        *op, std::shared_ptr<const
                        sh::filesystem::Eurl> eurl) = 0
```

Get the content of a file.

```
bool canCreateDirectory (sh::filesystem::Operation *op, std::shared_ptr<const
                        sh::filesystem::Eurl> eurl) = 0
```

Returns if it is allowed to create subdirectories in a certain directory.

```
void createDirectory (sh::filesystem::Operation *op, std::shared_ptr<const
                        sh::filesystem::Eurl> eurl) = 0
```

Create a directory.

```
bool canCreateLink (sh::filesystem::Operation *op, std::shared_ptr<const
                        sh::filesystem::Eurl> eurl) = 0
```

Returns if it is allowed to create a link in a certain directory.

```
void createLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                        eurl, QString target) = 0
```

Create a link.

```
bool canCreateFile (sh::filesystem::Operation *op, std::shared_ptr<const
                        sh::filesystem::Eurl> eurl) = 0
```

Returns if it is allowed to create files in a certain directory.

```
void createFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                        eurl, QIODevice *content, HandlerTransfer *handlertransfer) = 0
```

Creates a file with some content.

It may use handlertransfer for some better integration, like cancel support and progress monitoring.

```
bool canDeleteItem (sh::filesystem::Operation *op, std::shared_ptr<const
                        sh::filesystem::Eurl> eurl) = 0
```

Returns if it is allowed to delete a certain file/directory/link/...

void **deleteItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*>  
                  *eurl*) = 0  
Delete a file/directory/link/...

void **deleteDirectoryIfEmpty** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
                                  *sh::filesystem::Eurl*> *eurl*)  
Deletes a directory only if it is empty.

bool **canRenameItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
                                  *sh::filesystem::Eurl*> *src*) = 0  
Returns if it is allowed to rename a certain file, directory, link, ... (see `renameItem`).

void **renameItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*>  
                  *src*, *QString* *destpath*) = 0  
Renames an item to another path.

It can be a simple filename change or also changes in the deeper path segments.

QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> **getActions** (const  
  QList<std::shared\_ptr<const  
  *sh::filesystem::Eurl*>>  
  *eurls*) = 0  
Returns a list of actions (see former example) for showing for certain files.

void **viewEnteredDirectory** (std::shared\_ptr<const *sh::filesystem::Eurl*>)  
Callback for preparing stuff when the view entered some directory (e.g. for enabling watchers).  
See also `viewLeftDirectory`.

void **viewLeftDirectory** (std::shared\_ptr<const *sh::filesystem::Eurl*>)  
Callback for preparing stuff when the view left some directory. See also `viewEnteredDirectory`.

void **search** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
                  *sh::filesystem::Eurl*> *eurl*, std::function<void> std::shared\_ptr<const  
                  *sh::filesystem::Eurl*>, QList<std::shared\_ptr<*sh::search::SearchCriterion*>>,  
                  *sh::filesystem::FilesystemNodeType* *ftype*  
> *addfile*, std::function<void> std::shared\_ptr<const *sh::filesystem::Eurl*>> *visitdir*,  
QList<std::shared\_ptr<*sh::search::SearchCriterion*>> *criteria*)  
Helps searching for files.

Override this method with a behavior identical to the default, if a specialized implementation can be much faster than the default one.

void **customizeUi** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> *node*, bool \**showSearch-*  
                  *Panel*)  
Creates a custom gui, which becomes part of the fileview.

*sh::filesystem::FilesystemModel* \***model** ()  
A convenience shortcut to the *sh::filesystem::FilesystemModel* (a singleton).

void **doShutdown** ()  
Executes singleton shutdown.

void **shutdown** ()  
Shutdown down this singleton.

bool **isAlive** ()  
Returns if this singleton is alive (true until its shutdown begins).

**class SharcFilesystemHandler** : public *sh::filesystem::FilesystemHandler*, public *sh::base::Singleton*  
#include <*sharcfilesystemhandler.h*> A filesystem handler for the sharc archives.

This is a brutally easy and inefficient archive format which exists mostly for testing.



## Public Functions

```

void itemList (sh::filesystem::Operation *op, std::shared_ptr<const
              sh::filesystem::Eurl> eurl, sh::filesystem::FilesystemNodeType type,
              sh::filesystem::FilesystemNodeListEditor *list) override

void configureItems (sh::filesystem::Operation *op, QList<std::shared_ptr<sh::filesystem::FilesystemNode>>
                    nodes) override

sh::filesystem::FilesystemNodeType getType (sh::filesystem::Operation *op,
                                           std::shared_ptr<const sh::filesystem::Eurl>
                                           eurl) override

qint64 getSize (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
               eurl) override

QDateTime getMtime (sh::filesystem::Operation *op, std::shared_ptr<const
                  sh::filesystem::Eurl> eurl) override

void setMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
              eurl, QDateTime time) override

QString getMimeType (sh::filesystem::Operation *op, std::shared_ptr<const
                   sh::filesystem::Eurl> eurl) override

QString getLinkTarget (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) override

bool canGetFileContent (sh::filesystem::Operation *op, std::shared_ptr<const
                      sh::filesystem::Eurl> eurl) override

std::shared_ptr<QIODevice> getFileContent (sh::filesystem::Operation
                                         *op, std::shared_ptr<const
                                         sh::filesystem::Eurl> eurl) override

bool canCreateDirectory (sh::filesystem::Operation *op, std::shared_ptr<const
                       sh::filesystem::Eurl> eurl) override

void createDirectory (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) override

bool canCreateLink (sh::filesystem::Operation *op, std::shared_ptr<const
                  sh::filesystem::Eurl> eurl) override

void createLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                eurl, QString target) override

bool canCreateFile (sh::filesystem::Operation *op, std::shared_ptr<const
                  sh::filesystem::Eurl> eurl) override

void createFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                eurl, QIODevice *content, HandlerTransfer *handlertransfer) override

bool canDeleteItem (sh::filesystem::Operation *op, std::shared_ptr<const
                  sh::filesystem::Eurl> eurl) override

void deleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                eurl) override

bool canRenameItem (sh::filesystem::Operation *op, std::shared_ptr<const
                  sh::filesystem::Eurl> src) override

void renameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                src, QString destpath) override

```

```
QList<std::shared_ptr<sh::actions::AbstractActionItem>> getActions (const
                                                                    QList<std::shared_ptr<const
                                                                    sh::filesystem::Eurl>>
                                                                    eurls) override

void itemlist (sh::filesystem::Operation *op, std::shared_ptr<const
              sh::filesystem::Eurl> eurl, sh::filesystem::FilesystemNodeType type,
              sh::filesystem::FilesystemNodeListEditor *list) = 0
    Determine a list of subelements in a certain directory.

void configureItems (sh::filesystem::Operation *op, QList<std::shared_ptr<FilesystemNode>>
                    nodes)
    Configure newly created nodes (e.g. setting another icon or changing the display name).

sh::filesystem::FilesystemNodeType getType (sh::filesystem::Operation *op,
                                             std::shared_ptr<const sh::filesystem::Eurl>
                                             eurl) = 0
    Determine if a file is regular, or a dir, or a link, ...

qint64 getSize (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
               eurl) = 0
    Determine the size of a file.

QDateTime getMtime (sh::filesystem::Operation *op, std::shared_ptr<const
                   sh::filesystem::Eurl> eurl) = 0
    Determine the mtime of a file.

void setMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
               eurl, QDateTime time) = 0
    Set the mtime of a file.

QString getMimetype (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) = 0
    Determine mime type of a file.

QString getLinkTarget (sh::filesystem::Operation *op, std::shared_ptr<const
                      sh::filesystem::Eurl> eurl) = 0
    Resolve a link.

QList<QString> listExtendedAttributes (sh::filesystem::Operation *op,
                                       std::shared_ptr<const sh::filesystem::Eurl>
                                       eurl)
    Fetches the list of available extended attributes for an entry.

quint64 getExtendedAttributeSize (sh::filesystem::Operation *op,
                                   std::shared_ptr<const sh::filesystem::Eurl>
                                   eurl, QString attribute)
    Returns the size of value for one extended attribute for an entry.

QByteArray getExtendedAttribute (sh::filesystem::Operation *op,
                                  std::shared_ptr<const sh::filesystem::Eurl>
                                  eurl, QString attribute)
    Returns the value for one extended attribute for an entry.

void setExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
                           sh::filesystem::Eurl> eurl, QString attribute, QByteArray
                           value)
    Sets the value for one extended attribute for an entry.

void removeExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
                              sh::filesystem::Eurl> eurl, QString attribute)
    Removes one extended attributes for an entry.
```

QMap<QString, QString> **getCustomAttributes** (*sh::filesystem::Operation*  
*\*op*, *std::shared\_ptr<const*  
*sh::filesystem::Eurl> eurl*)

Returns the custom attributes for an entry.

In contrast to extended attributes, the custom attributes are not directly stored in the filesystem this way, but handle implementation specific aspects (like permissions).

void **setCustomAttribute** (*sh::filesystem::Operation* *\*op*, *std::shared\_ptr<const*  
*sh::filesystem::Eurl> eurl*, *QString attribute*, *QString value*)

Sets the value for one custom attribute for an entry.

See also getCustomAttributes.

bool **canGetFileContent** (*sh::filesystem::Operation* *\*op*, *std::shared\_ptr<const*  
*sh::filesystem::Eurl> eurl*) = 0

Returns if it is allowed to get the content of a certain file.

*std::shared\_ptr<QIODevice>* **getFileContent** (*sh::filesystem::Operation*  
*\*op*, *std::shared\_ptr<const*  
*sh::filesystem::Eurl> eurl*) = 0

Get the content of a file.

bool **canCreateDirectory** (*sh::filesystem::Operation* *\*op*, *std::shared\_ptr<const*  
*sh::filesystem::Eurl> eurl*) = 0

Returns if it is allowed to create subdirectories in a certain directory.

void **createDirectory** (*sh::filesystem::Operation* *\*op*, *std::shared\_ptr<const*  
*sh::filesystem::Eurl> eurl*) = 0

Create a directory.

bool **canCreateLink** (*sh::filesystem::Operation* *\*op*, *std::shared\_ptr<const*  
*sh::filesystem::Eurl> eurl*) = 0

Returns if it is allowed to create a link in a certain directory.

void **createLink** (*sh::filesystem::Operation* *\*op*, *std::shared\_ptr<const* *sh::filesystem::Eurl>*  
*eurl*, *QString target*) = 0

Create a link.

bool **canCreateFile** (*sh::filesystem::Operation* *\*op*, *std::shared\_ptr<const*  
*sh::filesystem::Eurl> eurl*) = 0

Returns if it is allowed to create files in a certain directory.

void **createFile** (*sh::filesystem::Operation* *\*op*, *std::shared\_ptr<const* *sh::filesystem::Eurl>*  
*eurl*, *QIODevice \*content*, *HandlerTransfer \*handlertransfer*) = 0

Creates a file with some content.

It may use *handlertransfer* for some better integration, like cancel support and progress monitoring.

bool **canDeleteItem** (*sh::filesystem::Operation* *\*op*, *std::shared\_ptr<const*  
*sh::filesystem::Eurl> eurl*) = 0

Returns if it is allowed to delete a certain file/directory/link/...

void **deleteItem** (*sh::filesystem::Operation* *\*op*, *std::shared\_ptr<const* *sh::filesystem::Eurl>*  
*eurl*) = 0

Delete a file/directory/link/...

void **deleteDirectoryIfEmpty** (*sh::filesystem::Operation* *\*op*, *std::shared\_ptr<const*  
*sh::filesystem::Eurl> eurl*)

Deletes a directory only if it is empty.

**bool canRenameItem** (*sh::filesystem::Operation* \*op, *std::shared\_ptr<const sh::filesystem::Eurl>* src) = 0  
Returns if it is allowed to rename a certain file, directory, link, ... (see `renameItem`).

**void renameItem** (*sh::filesystem::Operation* \*op, *std::shared\_ptr<const sh::filesystem::Eurl>* src, *QString destpath*) = 0  
Renames an item to another path.  
It can be a simple filename change or also changes in the deeper path segments.

**QList<std::shared\_ptr<sh::actions::AbstractActionItem>> getActions** (*const QList<std::shared\_ptr<const sh::filesystem::Eurl>> eurls*) = 0  
Returns a list of actions (see former example) for showing for certain files.

**bool requiresResolvedLinks** ()  
Returns if this handler requires to be used by the engine with resolved links.

**void viewEnteredDirectory** (*std::shared\_ptr<const sh::filesystem::Eurl>*)  
Callback for preparing stuff when the view entered some directory (e.g. for enabling watchers).  
See also `viewLeftDirectory`.

**void viewLeftDirectory** (*std::shared\_ptr<const sh::filesystem::Eurl>*)  
Callback for preparing stuff when the view left some directory. See also `viewEnteredDirectory`.

**bool isWellKnownScheme** ()  
Returns if the scheme for this handler is a well known one (which external programs typically would understand).

**void search** (*sh::filesystem::Operation* \*op, *std::shared\_ptr<const sh::filesystem::Eurl>* eurl, *std::function<void>* std::shared\_ptr<const sh::filesystem::Eurl>, *QList<std::shared\_ptr<sh::search::SearchCriterion>>*, *sh::filesystem::FilesystemNodeType* ftype, *addfile*, *std::function<void>* std::shared\_ptr<const sh::filesystem::Eurl>> visitdir, *QList<std::shared\_ptr<sh::search::SearchCriterion>>* criteria) Helps searching for files.  
Override this method with a behavior identical to the default, if a specialized implementation can be much faster than the default one.

**void customizeUi** (*std::shared\_ptr<sh::filesystem::FilesystemNode>* node, *bool \*showSearch-Panel*)  
Creates a custom gui, which becomes part of the fileview.

*sh::filesystem::FilesystemModel* \***model** ()  
A convenience shortcut to the *sh::filesystem::FilesystemModel* (a singleton).

**void doShutdown** ()  
Executes singleton shutdown.

**void shutdown** ()  
Shutdown down this singleton.

**bool isAlive** ()  
Returns if this singleton is alive (true until its shutdown begins).

## Public Static Functions

```
std::shared_ptr<const filesystem::Eurl> localToEurl (const QString local)
```

## Private Functions

```
SharcFilesystemHandler ()
```

```
class WindowsNetworkFilesystemHandler : public sh::filesystemhandlers::LocalFilesystemHandler
#include <windowsnetworkfilesystemhandler.h>
```

## Public Functions

```
void itemList (sh::filesystem::Operation *op, std::shared_ptr<const
sh::filesystem::Eurl> eurl, sh::filesystem::FilesystemNodeType type,
sh::filesystem::FilesystemNodeListEditor *list) override
```

```
void itemList (sh::filesystem::Operation *op, std::shared_ptr<const
sh::filesystem::Eurl> eurl, sh::filesystem::FilesystemNodeType type,
sh::filesystem::FilesystemNodeListEditor *list) = 0
```

Determine a list of subelements in a certain directory.

```
void configureItems (sh::filesystem::Operation *op, QList<std::shared_ptr<sh::filesystem::FilesystemNode>>
nodes) override
```

Configure newly created nodes (e.g. setting another icon or changing the display name).

```
sh::filesystem::FilesystemNodeType getType (sh::filesystem::Operation *op,
std::shared_ptr<const sh::filesystem::Eurl>
eurl) override
```

```
sh::filesystem::FilesystemNodeType getType (sh::filesystem::Operation *op,
std::shared_ptr<const sh::filesystem::Eurl>
eurl) = 0
```

Determine if a file is regular, or a dir, or a link, ...

```
qint64 getSize (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
eurl) override
```

```
qint64 getSize (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
eurl) = 0
```

Determine the size of a file.

```
QDateTime getMtime (sh::filesystem::Operation *op, std::shared_ptr<const
sh::filesystem::Eurl> eurl) override
```

```
QDateTime getMtime (sh::filesystem::Operation *op, std::shared_ptr<const
sh::filesystem::Eurl> eurl) = 0
```

Determine the mtime of a file.

```
void setMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
eurl, QDateTime mtime) override
```

```
void setMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
eurl, QDateTime time) = 0
```

Set the mtime of a file.

```
QString getMimeType (sh::filesystem::Operation *op, std::shared_ptr<const
sh::filesystem::Eurl> eurl) override
```

```
QString getMimeType (sh::filesystem::Operation *op, std::shared_ptr<const
sh::filesystem::Eurl> eurl) = 0
```

Determine mime type of a file.

```
QString getLinkTarget (sh::filesystem::Operation *op, std::shared_ptr<const  
                    sh::filesystem::Eurl> eurl) override
```

```
QString getLinkTarget (sh::filesystem::Operation *op, std::shared_ptr<const  
                    sh::filesystem::Eurl> eurl) = 0
```

Resolve a link.

```
bool canGetFileContent (sh::filesystem::Operation *op, std::shared_ptr<const  
                    sh::filesystem::Eurl> eurl) override
```

```
bool canGetFileContent (sh::filesystem::Operation *op, std::shared_ptr<const  
                    sh::filesystem::Eurl> eurl) = 0
```

Returns if it is allowed to get the content of a certain file.

```
std::shared_ptr<QIODevice> getFileContent (sh::filesystem::Operation  
                                         *op, std::shared_ptr<const  
                                         sh::filesystem::Eurl> eurl) override
```

```
std::shared_ptr<QIODevice> getFileContent (sh::filesystem::Operation  
                                         *op, std::shared_ptr<const  
                                         sh::filesystem::Eurl> eurl) = 0
```

Get the content of a file.

```
bool canCreateDirectory (sh::filesystem::Operation *op, std::shared_ptr<const  
                    sh::filesystem::Eurl> eurl) override
```

```
bool canCreateDirectory (sh::filesystem::Operation *op, std::shared_ptr<const  
                    sh::filesystem::Eurl> eurl) = 0
```

Returns if it is allowed to create subdirectories in a certain directory.

```
void createDirectory (sh::filesystem::Operation *op, std::shared_ptr<const  
                    sh::filesystem::Eurl> eurl) override
```

```
void createDirectory (sh::filesystem::Operation *op, std::shared_ptr<const  
                    sh::filesystem::Eurl> eurl) = 0
```

Create a directory.

```
bool canCreateLink (sh::filesystem::Operation *op, std::shared_ptr<const  
                    sh::filesystem::Eurl> eurl) override
```

```
bool canCreateLink (sh::filesystem::Operation *op, std::shared_ptr<const  
                    sh::filesystem::Eurl> eurl) = 0
```

Returns if it is allowed to create a link in a certain directory.

```
void createLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>  
                eurl, QString target) override
```

```
void createLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>  
                eurl, QString target) = 0
```

Create a link.

```
bool canCreateFile (sh::filesystem::Operation *op, std::shared_ptr<const  
                    sh::filesystem::Eurl> eurl) override
```

```
bool canCreateFile (sh::filesystem::Operation *op, std::shared_ptr<const  
                    sh::filesystem::Eurl> eurl) = 0
```

Returns if it is allowed to create files in a certain directory.

```
void createFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>  
                eurl, QIODevice *content, HandlerTransfer *handlertransfer) override
```

```
void createFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>  
                eurl, QIODevice *content, HandlerTransfer *handlertransfer) = 0
```

Creates a file with some content.



It may use `handlertransfer` for some better integration, like cancel support and progress monitoring.

```
bool canDeleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl) override
```

```
bool canDeleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl) = 0
```

Returns if it is allowed to delete a certain file/directory/link/...

```
void deleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl) override
```

```
void deleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl) = 0
```

Delete a file/directory/link/...

```
bool canRenameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> src) override
```

```
bool canRenameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> src) = 0
```

Returns if it is allowed to rename a certain file, directory, link, ... (see `renameItem`).

```
void renameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> src, QString destpath) override
```

```
void renameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> src, QString destpath) = 0
```

Renames an item to another path.

It can be a simple filename change or also changes in the deeper path segments.

```
QList<std::shared_ptr<sh::actions::AbstractActionItem>> getActions (const QList<std::shared_ptr<const sh::filesystem::Eurl>> eurls) override
```

```
QList<std::shared_ptr<sh::actions::AbstractActionItem>> getActions (const QList<std::shared_ptr<const sh::filesystem::Eurl>> eurls) = 0
```

Returns a list of actions (see former example) for showing for certain files.

```
QList<QString> listExtendedAttributes (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl) override
```

```
QList<QString> listExtendedAttributes (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl)
```

Fetches the list of available extended attributes for an entry.

```
quint64 getExtendedAttributeSize (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl, QString attribute) override
```

```
quint64 getExtendedAttributeSize (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl, QString attribute)
```

Returns the size of value for one extended attribute for an entry.

```
QByteArray getExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl, QString attribute) override
```

```
QByteArray getExtendedAttribute (sh::filesystem::Operation *op,
                                std::shared_ptr<const sh::filesystem::Eurl>
                                eurl, QString attribute)
```

Returns the value for one extended attribute for an entry.

```
void setExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
sh::filesystem::Eurl> eurl, QString attribute, QByteArray
value) override
```

```
void setExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
sh::filesystem::Eurl> eurl, QString attribute, QByteArray
value)
```

Sets the value for one extended attribute for an entry.

```
void removeExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
sh::filesystem::Eurl> eurl, QString attribute)
override
```

```
void removeExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
sh::filesystem::Eurl> eurl, QString attribute)
```

Removes one extended attributes for an entry.

```
QMap<QString, QString> getCustomAttributes (sh::filesystem::Operation
*op, std::shared_ptr<const
sh::filesystem::Eurl> eurl) override
```

```
QMap<QString, QString> getCustomAttributes (sh::filesystem::Operation
*op, std::shared_ptr<const
sh::filesystem::Eurl> eurl)
```

Returns the custom attributes for an entry.

In contrast to extended attributes, the custom attributes are not directly stored in the filesystem this way, but handle implementation specific aspects (like permissions).

```
void setCustomAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
sh::filesystem::Eurl> eurl, QString attribute, QString value)
override
```

```
void setCustomAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
sh::filesystem::Eurl> eurl, QString attribute, QString value)
```

Sets the value for one custom attribute for an entry.

See also getCustomAttributes.

```
bool requiresResolvedLinks () override
```

Returns if this handler requires to be used by the engine with resolved links.

```
bool isWellKnownScheme () override
```

Returns if the scheme for this handler is a well known one (which external programs typically would understand).

```
QString eurlToLocal (std::shared_ptr<const sh::filesystem::Eurl> eurl)
```

```
std::shared_ptr<const sh::filesystem::Eurl> localToEurl (const QString local)
```

```
void deleteDirectoryIfEmpty (sh::filesystem::Operation *op, std::shared_ptr<const
sh::filesystem::Eurl> eurl)
```

Deletes a directory only if it is empty.

```
void viewEnteredDirectory (std::shared_ptr<const sh::filesystem::Eurl>)
```

Callback for preparing stuff when the view entered some directory (e.g. for enabling watchers). See also viewLeftDirectory.

```
void viewLeftDirectory (std::shared_ptr<const sh::filesystem::Eurl>)
```

Callback for preparing stuff when the view left some directory. See also viewEnteredDirectory.



```
void search (sh::filesystem::Operation *op, std::shared_ptr<const
sh::filesystem::Eurl> eurl, std::function<void> std::shared_ptr<const
sh::filesystem::Eurl>, QList<std::shared_ptr<sh::search::SearchCriterion>>,
sh::filesystem::FilesystemNodeType ftype
> addfile, std::function<void std::shared_ptr<const sh::filesystem::Eurl>> visitdir,
QList<std::shared_ptr<sh::search::SearchCriterion>> criteria
```

Helps searching for files.

Override this method with a behavior identical to the default, if a specialized implementation can be much faster than the default one.

```
void customizeUi (std::shared_ptr<sh::filesystem::FilesystemNode> node, bool *showSearch-
Panel)
```

Creates a custom gui, which becomes part of the fileview.

```
sh::filesystem::FilesystemModel *model ()
```

A convenience shortcut to the *sh::filesystem::FilesystemModel* (a singleton).

```
void doShutdown ()
```

Executes singleton shutdown.

```
void shutdown ()
```

Shutdown down this singleton.

```
bool isAlive ()
```

Returns if this singleton is alive (true until its shutdown begins).

## namespace **paneldetails**

Infrastructure for panel details.

Can be shown in the bottom part of the main window. Beyond the infrastructure, it also includes some very basic detail providers.

See *sh::paneldetails::PanelDetailManager* for more.

## Enums

```
enum PanelDetailPositionIndex
```

Base values for position indexes of panel details.

Values:

```
enumerator VeryInteresting = 1000000
```

```
enumerator Interesting = 2000000
```

```
enumerator Exotic = 3000000
```

```
class CommonPanelDetails
```

#include <commonpaneldetails.h> Implementations of common details provider for the details panel.

## Public Static Functions

```
void doInitialize ()
```

```
void doShutdown ()
```

```
void _ViaDetailColumn (std::shared_ptr<PanelDetail> detail,
std::shared_ptr<sh::filesystem::FilesystemNode> node,
std::shared_ptr<sh::filesystem::DetailColumn> column, QString
keyname, std::function<QString> QVariant
> valtostring
```

```
void NumberOfSelectedItems (std::shared_ptr<PanelDetail>,
                             QList<std::shared_ptr<sh::filesystem::FilesystemNode>>,
                             sh::filesystem::Operation *op)

void TotalFileSize (std::shared_ptr<PanelDetail>, QList<std::shared_ptr<sh::filesystem::FilesystemNode>>,
                    sh::filesystem::Operation *op)

void FileType (std::shared_ptr<PanelDetail> detail, std::shared_ptr<sh::filesystem::FilesystemNode>
               node, sh::filesystem::Operation *op)

void FileSize (std::shared_ptr<PanelDetail> detail, std::shared_ptr<sh::filesystem::FilesystemNode>
               node, sh::filesystem::Operation *op)

void FileModificationTime (std::shared_ptr<PanelDetail> detail,
                             std::shared_ptr<sh::filesystem::FilesystemNode> node,
                             sh::filesystem::Operation *op)
```

**class PanelDetail** : public QObject

*#include <paneldetailmanager.h>* A detail panel entry.

Is essentially a list of *PanelDetailRow* and some layout information.

### Public Functions

**PanelDetail** (int *positionIndex*, int *valueWidthHint*)

Constructed only by the infrastructure and made available otherwise.

QList<*PanelDetailRow*> **rows** ()

Returns the list of detail rows stored in this panel detail.

int **positionIndex** ()

Returns a position index.

It controls the position of this panel detail in relation to the other ones.

int **valueWidthHint** ()

Returns the specified width for the panel detail.

void **setRows** (QList<*PanelDetailRow*> *rows*)

Sets the list of detail rows.

void **addRow** (*PanelDetailRow* *row*)

Adds a new detail row.

void **addOrReplaceRow** (*PanelDetailRow* *row*)

Adds a new detail row (replacing an old one with the same key).

void **\_emit\_linkTriggered** (QString *link*)

### Signals

void **changed** ()

Emitted when data has changed.

void **linkTriggered** (QString *link*)

Emitted when a link is triggered by the user.

## Private Members

QMutex **\_rowsmutex**  
 QList<*PanelDetailRow*> **\_rows**  
 int **\_positionIndex**  
 int **\_valueWidthHint**

### class PanelDetailFactory

*#include <paneldetailmanager.h>* Abstract factory for creating panel details for the selected nodes.

Subclassed by *sh::paneldetails::PanelDetailFactoryForFunctionMulti*,  
*sh::paneldetails::PanelDetailFactoryForFunctionSingle*

## Public Functions

std::shared\_ptr<*PanelDetail*> **construct** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>>  
*nodes*, *sh::filesystem::Operation \*op*) = 0

### class PanelDetailFactoryForFunctionMulti : public sh::paneldetails::PanelDetailFactory

*#include <paneldetailmanager.h>* A factory for creating panel details for more than one selected node.

Subclassed by *sh::scripting::api::ApiPanelDetailFactoryMulti*

## Public Functions

**PanelDetailFactoryForFunctionMulti** (std::function<void> std::shared\_ptr<*PanelDetail*>,  
 QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>>,  
*sh::filesystem::Operation \*op*  
 > *fcnsetvalue*, std::function<void> QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>>,  
 QString link> *fcnlinktriggered*, int *position*, int *valueWidthHint* Constructed only indirectly.

std::shared\_ptr<*PanelDetail*> **construct** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>>  
*nodes*, *sh::filesystem::Operation \*op*)

### class PanelDetailFactoryForFunctionSingle : public sh::paneldetails::PanelDetailFactory

*#include <paneldetailmanager.h>* A factory for creating panel details for just one selected node.

Subclassed by *sh::scripting::api::ApiPanelDetailFactorySingle*

## Public Functions

**PanelDetailFactoryForFunctionSingle** (std::function<void> std::shared\_ptr<*PanelDetail*>,  
 std::shared\_ptr<*sh::filesystem::FilesystemNode*>,  
*sh::filesystem::Operation \*op*  
 > *fcnsetvalue*, std::function<void> std::shared\_ptr<*sh::filesystem::FilesystemNode*>, QString link>  
*fcnlinktriggered*, int *position*, int *valueWidthHint* Constructed only indirectly.

std::shared\_ptr<*PanelDetail*> **construct** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>>  
*nodes*, *sh::filesystem::Operation \*op*)

### class PanelDetailManager : public QObject, public sh::base::Singleton

*#include <paneldetailmanager.h>* Manager for details provider as used in the details panel.

Register some code and data here for implementing a new provider.

## Public Functions

`QList<std::shared_ptr<PanelDetail>> getDetails (QList<std::shared_ptr<sh::filesystem::FilesystemNode>> nodes)`

Returns a list of *PanelDetail* for a list of selected filesystem nodes.

`std::shared_ptr<PanelDetailFactory> registerFactorySingle (std::shared_ptr<PanelDetailFactory> factory)`

Registers a panel detail factory for single selection by *PanelDetailFactory* instance.

`std::shared_ptr<PanelDetailFactory> registerFactoryMulti (std::shared_ptr<PanelDetailFactory> factory)`

Registers a panel detail factory for multi selection by *PanelDetailFactory* instance.

`std::shared_ptr<PanelDetailFactory> registerFactorySingle (std::function<void> std::shared_ptr<PanelDetail> std::shared_ptr<sh::filesystem::FilesystemNode> sh::filesystem::Operation*`

`> fctsetvalue, std::function<void>std::shared_ptr<sh::filesystem::FilesystemNode>, QString link> fctlinktriggered, int position, int valueWidthHint`Registers a panel detail factory for single selection by parameters.

`std::shared_ptr<PanelDetailFactory> registerFactoryMulti (std::function<void> std::shared_ptr<PanelDetail> QList<std::shared_ptr<sh::filesystem::FilesystemNode>> sh::filesystem::Operation*`

`> fctsetvalue, std::function<void>QList<std::shared_ptr<sh::filesystem::FilesystemNode>>, QString link> fctlinktriggered, int position, int valueWidthHint`Registers a panel detail factory for multi selection by parameters.

`void doInitialize ()`

Executes singleton initialization.

`void doShutdown ()`

Executes singleton shutdown.

`void shutdown ()`

Shutdown down this singleton.

`bool isAlive ()`

Returns if this singleton is alive (true until its shutdown begins).

## Private Functions

`PanelDetailManager ()`

## Private Members

`QList<std::shared_ptr<PanelDetailFactory>> _factoriesingle`

`QList<std::shared_ptr<PanelDetailFactory>> _factoriesmulti`

`QMutex _mutex`

**class PanelDetailRow**

*#include <paneldetailmanager.h>* One pair of label text and value text in a panel detail.

## Public Functions

**PanelDetailRow** (QString *key*, QList<*PanelDetailRowValueElement*> *value*)

Is intended to be directly constructed from everywhere.

**PanelDetailRow** (QList<*PanelDetailRowValueElement*> *value*)

Is intended to be directly constructed from everywhere.

QString **key** ()

QList<*PanelDetailRowValueElement*> **value** ()

## Private Members

QString **\_key**

QList<*PanelDetailRowValueElement*> **\_value**

**class PanelDetailRowValueElement**

*#include <paneldetailmanager.h>* One element of a detail row's value.

This can be something like a string, an icon or a link button.

## Public Functions

**PanelDetailRowValueElement** ()

A placeholder for a value which is not yet arrived. It could be visualized with a loading animation or '...'. Is intended to be directly constructed from everywhere.

**PanelDetailRowValueElement** (QString *text*)

A piece of text. Is intended to be directly constructed from everywhere.

**PanelDetailRowValueElement** (QIcon *icon*)

An icon. Is intended to be directly constructed from everywhere.

**PanelDetailRowValueElement** (QString *text*, QString *linktarget*, bool *autohide*)

A link button. Is intended to be directly constructed from everywhere.

QString **text** ()

Returns the text.

QIcon **icon** ()

Returns the icon.

QString **linktarget** ()

Returns the link target.

bool **linkautohide** ()

Returns if this link shall automatically hide (e.g. when the mouse disappears).

bool **isWaiting** ()

Returns if it is currently waiting for an update.

### Private Members

```
QString _text
QIcon _icon
QString _linktarget
bool _linkautohide = false
bool _iswaiting = false
```

### namespace scripting

The Shallot scripting interface.

### Typedefs

```
using StringMap = QMap<QString, V>
```

### class Api

*#include <api.h>* Interoperation layer between the Shallot core and a plugin interpreter.

### Public Functions

```
Api ()
~Api ()
QStringList classes ()
QStringList rootMembers ()
ApiClass *getClass (QString name)
ApiClass *getClassByIdName (QString typeidName)
void *rootMember (QString name)
ApiClass *rootMemberClass (QString name)
```

### Private Functions

```
void registerClass (QString name, ApiClass **apiclass, const std::type_info &nativeType)
void registerMethod (ApiClass *apiclass, QString name)
void registerRootObjectMember (QString name, ApiClass *cls, void *member)
```

### Private Members

```
QHash<QString, ApiClass*> _classes
QHash<QString, ApiClass*> _classesByNativeType
QHash<QString, void*> _rootMembers
QHash<QString, ApiClass*> _rootMembersClasses
class ApiClass
#include <api.h> Data structure for one api-aware core class.
```

## Public Functions

**ApiClass** (QString *name*)  
Constructed only indirectly.

**~ApiClass** ()

QString **name** ()

QStringList **methods** ()

void **addMethod** (*ApiMethod* \**m*)

*ApiMethod* \***getMethod** (QString *name*)

## Private Members

QString **\_name**

QHash<QString, *ApiMethod*\*> **\_methods**

**class ApiMethod**

*#include <api.h>* Data structure for one api-aware core method.

## Public Functions

**ApiMethod** (QString *name*)  
Constructed only indirectly.

QString **name** ()

**class PythonScriptInterpreter** : public *sh::scripting::ScriptInterpreter*, public *sh::base::Singleton*  
*#include <pythonscriptinterpreter.h>* Integration of the Python language for scripting.

Find also the scripting manual for all details about the scripting interface.

## Public Functions

**~PythonScriptInterpreter** ()

bool **isAvailable** ()

void **initializeInterpreter** ()

QString **filesuffix** ()

void **execute** (QString *script*)

void **doInitialize** ()  
Executes singleton initialization.

void **shutdown** ()  
Shutdown down this singleton.

bool **isAlive** ()  
Returns if this singleton is alive (true until its shutdown begins).

### Public Members

```
PyObject * _Exception
PyObject * _CancelException
PyObject * _ThreadAbortException
```

### Public Static Functions

```
void registerMethod (sh::scripting::ApiClass *cls, QString name, PyCFunction fctptr)
void throwNativeExceptionFromPythonTraceback ()
```

### Public Static Attributes

```
QHash<void*, PyObject*> _buddies
QHash<void*, std::weak_ptr<void>> _natives
```

### Private Functions

```
PythonScriptInterpreter ()
```

### Private Static Functions

```
PyObject *getApiModuleDef ()
```

### Private Static Attributes

```
void *apiModuleDef
sh::scripting::Api *api = new sh::scripting::Api()
QHash<QString, PyCFunction> _methods
PyObject *modtraceback

class ScriptingEngine : public QObject, public sh::base::Singleton
    #include <scriptingengine.h> The Scripting engine loads scripts by means of the registered inter-
    preters.
```

### Public Functions

```
void loadScript (QString script)
    Loads a plugin script from file.

void doShutdown ()
    Executes singleton shutdown.

void shutdown ()
    Shutdown down this singleton.

bool isAlive ()
    Returns if this singleton is alive (true until its shutdown begins).
```



## Private Functions

**ScriptingEngine** ()

void **registerInterpreter** (std::shared\_ptr<*sh::scripting::ScriptInterpreter*> *interpreter*)

## Private Members

QMap<QString, std::shared\_ptr<*sh::scripting::ScriptInterpreter*>> **\_interpreters**

## Private Static Attributes

std::shared\_ptr<*sh::configuration::ConfigurationValue*> **cfgvallastStartupFailed** = *sh::configuration::Config*

**class ActionPluginLoadCrashedAgain** : public *sh::actions::ActionActionItem*

## Public Functions

**ActionPluginLoadCrashedAgain** (QString *path*, QMutex \**mutex*)

void **execute** ()

Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)

Executes this action.

QKeySequence **shortcuthint** ()

Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()

Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)

Sets the keyboard shortcut for triggering this action.

QString **text** ()

Returns the displayed text for this action.

QString **icon** ()

Returns the icon for this action.

bool **enabled** ()

Checks if this action is enabled.

bool **isChecked** ()

Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () const

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See *ActionDefaultPrecedenceValues* for reference values.

```
void setText (QString text)  
    Sets the displayed text.  
  
void setIcon (QString icon)  
    Sets the icon.  
  
void setEnabled (bool enabled)  
    Sets if the item is enabled.  
  
void setChecked (bool checked)  
    Sets if the item is checked (has a cross in the ui).  
  
void setVisible (bool visible)  
    Sets the visibility of this item.  
  
bool visible ()  
    Checks the visibility of this item (non-recursively).  
  
std::weak_ptr<AbstractActionItem> parentAction ()  
    Returns the parent action, if it is added to a container.  
  
void _setParentAction (std::shared_ptr<AbstractActionItem> parent)  
    Sets the parent action. .  
  
void initializeAsync (std::function<void>  
    > oninitialized = 0) Asynchronously initializes the action.  
  
void initializeSync ()  
    Synchronously initializes the action.  
  
bool isInitialized ()  
    Checks if this action is initialized.  
  
bool isInitializing ()  
    Checks if this action is initializing.
```

## Signals

```
void changed ()  
    Emits when some data changed.
```

```
class ScriptedException : public sh::exceptions::Exception  
    #include <scriptingengine.h> Exception for errors within the scripting engine and interpreters.
```

## Public Functions

```
ScriptedException (sh::exceptions::ExceptionData data)  
  
QString message () const  
  
QString name () const  
  
QString classes () const  
  
QString details () const  
  
QString callstack () const  
  
QString auxiliary () const  
  
QString customValue (QString key) const  
  
bool isRuntimeException () const
```

```

bool isProgramException () const
bool isRetryable () const
bool isResumeable () const
bool isDetailsAreInteresting () const
int autoRetryRecommendedIn () const
void setResumeable (bool v)
void setReTryable (bool v)
void setCustomValue (QString key, QString value)
bool isClass (QString classname)
sh::exceptions::ExceptionData data ()

```

### Public Static Functions

```

template<class Handler>
std::shared_ptr<RegisterHandler<Handler>> createRegisterHandler (Handler
                                                                handler, QS-
                                                                tack<Handler>
                                                                *stack)

void executeGuarded (std::function<void>
    > fctint flags = 0)Executes some code with some standard exection handling around.

Parameters
    • fct: The code to execute guarded.
    • flags: Flags of ExecuteGuardFlag for choosing the behavior.

void executeGuarded_errorpanel (std::function<void>
    > fctint flags = 0

QString _demangle (QString l)

void exceptionDialog (sh::exceptions::Exception &e, bool *doRetry)

```

### Public Static Attributes

```

const QString UNDEFINED_ERROR_OCCURRED = QObject::tr("An unspecified error occurred.")
const QString SHALLOT_MUST_CLOSE_TEXT = QObject::tr("Your Shallot process is disturbed by an error and")
const QString Value_isShallotException = "_isShallotException"

```

**class ScriptInterpreter**

*#include <scriptingengine.h>* The base class for integration of a script language.

Subclassed by *sh::scripting::PythonScriptInterpreter*

### Public Functions

```
bool isAvailable ()  
void initializeInterpreter ()  
QString filesuffix ()  
void execute (QString)  
~ScriptInterpreter ()
```

```
struct shallot_ShallotObject
```

A wrapper around a native object for the Python world.

### Public Members

```
PyObject_HEAD void * nativeObject  
std::shared_ptr<void> sharedNativeObject  
bool isReallyShared
```

```
namespace api
```

Adapter functions and classes for interaction with scripting.

Some programming interfaces in the core are not directly usable for the scripting api. Scripting-friendly surrogates for them are here.

```
class ApiActionActionItem: public sh::actions::ActionActionItem  
    #include <apiaction.h>
```

### Public Functions

```
ApiActionActionItem (QString text, bool enabled, QString icon, int defaultActionPrecedence)
```

```
void _execute ()
```

```
void _initializeSync ()
```

```
void action (sh::actions::ActionExecutionInfo *info) override
```

The action implementation, i.e. what the actions should actually do.

```
void initialize () override
```

Initialize the action. This should make the time-consuming parts, e.g. for determining a label or enabled state.

```
void execute ()
```

Executes this action.

```
void execute (sh::actions::ActionExecutionInfo *info)
```

Executes this action.

```
QKeySequence shortcuthint ()
```

Returns the keyboard shortcut for triggering this action.

```
bool shortcuthintTriggersOnCurrentDirectory ()
```

Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcutHint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<AbstractActionItem> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<AbstractActionItem> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Public Members

```
std::function<void ()> _initialize  
std::function<void (sh::actions::ActionExecutionInfo*)> _action
```

## Signals

```
void changed ()  
    Emits when some data changed.
```

```
class ApiActionFactory : public sh::actions::AbstractActionFactory  
    #include <apipredicatedactionfactory.h>
```

## Public Functions

```
ApiActionFactory (QString category, QList<std::shared_ptr<actions::Predicate>> predicates)  
std::shared_ptr<sh::actions::AbstractActionItem> construct (QList<std::shared_ptr<sh::filesystem::FilesystemNode>> nodes)  
std::shared_ptr<ActionInstantiation> actionAvailable (ActionInstantiation *instantiation)
```

## Public Members

```
std::function< std::shared_ptr< sh::actions::AbstractActionItem >QList< std::shared_ptr< sh::actions::AbstractActionItem >> >  
class ApiDetailColumn : public sh::filesystem::DetailColumn  
    #include <apidetailcolumn.h>
```

## Public Functions

```
ApiDetailColumn (QString name, QString displayName, int displayIndex, bool sort_doTypediff, int defaultWidth, bool isRightAligned)  
QVariant determineValue (std::shared_ptr<sh::filesystem::FilesystemNode> node, sh::filesystem::Operation *op)  
void applyValue (std::shared_ptr<const sh::filesystem::Eurl> eurl, sh::filesystem::Operation *op, QVariant value)  
QString name ()  
    The internal unique name.  
QString displayName ()  
    The display name.  
bool isValueAvailable (std::shared_ptr<FilesystemNode> node)  
    Checks if a value for this detail is available for one given node.  
QVariant value (std::shared_ptr<FilesystemNode> node, bool ignoreAged = false)  
    Returns the value for this detail for one given node.
```

### Parameters

- ignoreAged: If no value should be returned which is older than the last request (not useful for nearly all cases).

QString **displayValue** (std::shared\_ptr<FilesystemNode> *node*, const  
*sh::filesystem::FilesystemModelFileviewProxy* \*viewmodel)

Returns the stringified value for this details for one given node considering the configuration of a view.

bool **isVisible** ()

If this detail shall be a visible column in the view.

QVariant **requestValue** (std::shared\_ptr<FilesystemNode> *node*,  
*sh::filesystem::Operation* \*op = 0, bool *withNodeValues* =  
false, bool *withOperationsCache* = true)

Requests to determine the value for this detail for one given node. Blocks until the value is available. Get it with the value method afterwards.

bool **sort\_doTypediff** ()

If sorting should separate files and directories.

int **sort\_order** (std::shared\_ptr<FilesystemNode> *n1*, std::shared\_ptr<FilesystemNode>  
*n2*)

The sorting order.

uint **displayIndex** ()

Controls which place this column should get in the user interface.

QVariant **computeValue** (std::shared\_ptr<FilesystemNode> *node*,  
*sh::filesystem::Operation* \*op)

Returns a freshly determined value for this column and the given node. It neither asks nor populates any caches or storages.

int **defaultWidth** ()

bool **isRightAligned** ()

void **applyValueByEurl** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*,  
*sh::filesystem::Operation* \*op, QVariant *value*)

Applies the detail value to a given eurl (without using any caches).

Used for transferring some details when a file transfer occurs.

Optionally implement the one of the applyValueBy... methods and register the instance with *sh::filesystem::FilesystemOperation::addTransferrableDetailColumn*. For performance reasons, you should decide to implement applyValueByEurl.

void **applyValueByNode** (std::shared\_ptr<FilesystemNode> *node*,  
*sh::filesystem::Operation* \*op, QVariant *value*)

Applies the detail value to a given node (without using any caches).

Used for transferring some details when a file transfer occurs.

Optionally implement the one of the applyValueBy... methods and register the instance with *sh::filesystem::FilesystemOperation::addTransferrableDetailColumn*. For performance reasons, you should decide to implement applyValueByEurl.

## Public Members

```
std::function<QString (std::shared_ptr<sh::filesystem::FilesystemNode>,
                        sh::filesystem::Operation*)> _determineValue
std::function<void (std::shared_ptr<sh::filesystem::Eurl>,          sh::filesystem::Operation*,
                    QString)> _applyValue
```

## Public Static Functions

```
QVariant VALUE_UNAVAILABLE ()
    A special value expressing that the value is not yet available.
void registerKnownDetailColumn (QString name, std::shared_ptr<DetailColumn>
                                column)
std::shared_ptr<DetailColumn> findKnownDetailColumn (QString name)
```

## Public Static Attributes

```
const uint DISPLAYINDEX_CORE = 10000
const uint DISPLAYINDEX_INTERESTING = 20000
const uint DISPLAYINDEX_EXOTIC = 30000
class ApiFilePropertyDialogTab: public sh::ui::FilePropertyDialogTab
    #include <apifilepropertydialogtab.h>
```

## Public Functions

```
ApiFilePropertyDialogTab (QString title, QList<PropertyConfig> properties)
QString title () override
    The tab title. .
QList<QString> properties () override
    Returns the list of property labels (one entry for each widget).
    Each property is displayed with a header and a widget (created in createWidget) with some
    content (populated in updateWidget).
void populateWidget (int i, sh::ui::FilePropertyDialogTabActionsView *widget)
    override
    Populates the tab widget for the i-th property.
    Typically uses the FilePropertyDialog::create* methods to create sub-widgets.
    See also dialog().
void updateWidget (int i, sh::ui::FilePropertyDialogTabActionsView *widget,
                   sh::filesystem::Operation *op) override
    Populates the widget for the i-th property with actual content. .
QString titleWithoutMnemonic ()
    The tab title without mnemonic.
void refresh ()
    Refreshes the complete content.
```



Typically called after some user actions in a property widget require that all the other content must be refreshed.

`QList<std::shared_ptr<sh::filesystem::FilesystemNode>> nodes ()`

The nodes to show.

`FilePropertyDialogTabActionsView *widgetAt (int i)`

Returns the widget for the i-th property.

`int widgetCount ()`

Returns the number of widgets.

`std::shared_ptr<FilePropertyDialog> dialog ()`

Returns the associated dialog.

## Public Members

`std::function< QList< QString >int, sh::filesystem::Operation *, qintptr>> _u`

`std::function<void (QString)> _buttonTriggered`

## Public Static Attributes

`const int PROPERTY_TYPE_STRING = 1`

`const int PROPERTY_TYPE_STRINGMAP = 2`

`const int PROPERTY_TYPE_ICONTEXTBANNER = 3`

## Private Types

`typedef QPair<QString, QString> StringPair`

## Private Members

`QString _title`

`QList<PropertyConfig> _properties`

`class PropertyConfig`

`#include <apifilepropertydialogtab.h>`

## Public Types

`typedef QPair<QString, QString> ButtonConfig`

### Public Functions

**PropertyConfig** (QString *title*, int *propertytype*, QList<*ButtonConfig*> *buttons*)

### Public Members

QString **title**

int **propertytype**

QList<*ButtonConfig*> **buttons**

```
class ApiFilePropertyDialogTabFactory : public sh::ui::FilePropertyDialogTabFactory  
#include <apifilepropertydialogtab.h>
```

### Public Functions

**ApiFilePropertyDialogTabFactory** ()

std::shared\_ptr<*sh::ui::FilePropertyDialogTab*> **construct** (std::shared\_ptr<*sh::ui::FilePropertyDialog*>  
dialog)

std::shared\_ptr<*sh::ui::FilePropertyDialogTab*> **construct** (std::shared\_ptr<*sh::ui::FilePropertyDialog*>  
dialog) = 0

Constructs a fresh instance of a *FilePropertyDialogTab* implementation. .

### Public Members

std::function< std::shared\_ptr< *sh::scripting::api::ApiFilePropertyDialogTab*

### Public Static Attributes

const int **REGISTER\_FILEPROPERTYDIALOGTAB\_INDEX\_CORE** = 10000

Base value for display indexes of core (i.e. maximum important) tabs.

Used in *FilePropertyDialog::addTabFactory*.

const int **REGISTER\_FILEPROPERTYDIALOGTAB\_INDEX\_VERYIMPORTANT** = 20000

Base value for display indexes of very important tabs.

Used in *FilePropertyDialog::addTabFactory*.

const int **REGISTER\_FILEPROPERTYDIALOGTAB\_INDEX\_IMPORTANT** = 30000

Base value for display indexes of important tabs.

Used in *FilePropertyDialog::addTabFactory*.

const int **REGISTER\_FILEPROPERTYDIALOGTAB\_INDEX\_NORMAL** = 40000

Base value for display indexes of tabs with medium importance.

Used in *FilePropertyDialog::addTabFactory*.

const int **REGISTER\_FILEPROPERTYDIALOGTAB\_INDEX\_EXOTIC** = 50000

Base value for display indexes of exotic (i.e. less important) tabs.

Used in *FilePropertyDialog::addTabFactory*.

```
class ApiFilesystemHandler : public sh::filesystem::FilesystemHandler  
#include <apifilesystemhandler.h>
```

## Public Functions

**ApiFilesystemHandler** (*sh::filesystem::FilesystemModel* \*model)

void **itemlist** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl*> eurl, *sh::filesystem::FilesystemNodeType* type,  
*sh::filesystem::FilesystemNodeListEditor* \*list)

void **configureItems** (*sh::filesystem::Operation* \*op, QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>>  
nodes)

*sh::filesystem::FilesystemNodeType* **getType** (*sh::filesystem::Operation* \*op,  
std::shared\_ptr<const *sh::filesystem::Eurl*>  
eurl)

qint64 **getSize** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*>  
eurl)

QDateTime **getMtime** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl*> eurl)

void **setMtime** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*>  
eurl, QDateTime time)

QString **getMimeType** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl*> eurl)

QString **getLinkTarget** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl*> eurl)

QList<QString> **listExtendedAttributes** (*sh::filesystem::Operation*  
\*op, std::shared\_ptr<const  
*sh::filesystem::Eurl*> eurl)

quint64 **getExtendedAttributeSize** (*sh::filesystem::Operation* \*op,  
std::shared\_ptr<const *sh::filesystem::Eurl*>  
eurl, QString attribute)

QByteArray **getExtendedAttribute** (*sh::filesystem::Operation* \*op,  
std::shared\_ptr<const *sh::filesystem::Eurl*>  
eurl, QString attribute)

void **setExtendedAttribute** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl*> eurl, QString attribute, QByteArray  
value)

void **removeExtendedAttribute** (*sh::filesystem::Operation* \*op,  
std::shared\_ptr<const *sh::filesystem::Eurl*> eurl,  
QString attribute)

QMap<QString, QString> **getCustomAttributes** (*sh::filesystem::Operation*  
\*op, std::shared\_ptr<const  
*sh::filesystem::Eurl*> eurl)

void **setCustomAttribute** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl*> eurl, QString attribute, QString value)

bool **canGetFileContent** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl*> eurl)

std::shared\_ptr<QIODevice> **getFileContent** (*sh::filesystem::Operation*  
\*op, std::shared\_ptr<const  
*sh::filesystem::Eurl*> eurl)

bool **canCreateDirectory** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl*> eurl)

```

void createDirectory (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl)

bool canCreateLink (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl)

void createLink (sh::filesystem::Operation *op, std::shared_ptr<const
                 sh::filesystem::Eurl> eurl, QString target)

bool canCreateFile (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl)

void createFile (sh::filesystem::Operation *op, std::shared_ptr<const
                 sh::filesystem::Eurl> eurl, QIODevice *content, HandlerTransfer
                 *handlertransfer)

bool canDeleteItem (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl)

void deleteItem (sh::filesystem::Operation *op, std::shared_ptr<const
                 sh::filesystem::Eurl> eurl)

bool canRenameItem (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> src)

void renameItem (sh::filesystem::Operation *op, std::shared_ptr<const
                 sh::filesystem::Eurl> src, QString destpath)

QList<std::shared_ptr<sh::actions::AbstractActionItem>> getActions (const
                                                                QList<std::shared_ptr<const
                                                                sh::filesystem::Eurl>>
                                                                eurls)

void itemlist (sh::filesystem::Operation *op, std::shared_ptr<const
                 sh::filesystem::Eurl> eurl, sh::filesystem::FilesystemNodeType type,
                 sh::filesystem::FilesystemNodeListEditor *list) = 0
    Determine a list of subelements in a certain directory.

void configureItems (sh::filesystem::Operation *op, QList<std::shared_ptr<FilesystemNode>>
                     nodes)
    Configure newly created nodes (e.g. setting another icon or changing the display name).

sh::filesystem::FilesystemNodeType getType (sh::filesystem::Operation *op,
                                             std::shared_ptr<const sh::filesystem::Eurl>
                                             eurl) = 0
    Determine if a file is regular, or a dir, or a link, ...

qint64 getSize (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                 eurl) = 0
    Determine the size of a file.

QDateTime getMtime (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) = 0
    Determine the mtime of a file.

void setMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                 eurl, QDateTime time) = 0
    Set the mtime of a file.

QString getMimeType (sh::filesystem::Operation *op, std::shared_ptr<const
                     sh::filesystem::Eurl> eurl) = 0
    Determine mime type of a file.

QString getLinkTarget (sh::filesystem::Operation *op, std::shared_ptr<const
                       sh::filesystem::Eurl> eurl) = 0
    Resolve a link.

```

`QList<QString> listExtendedAttributes (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl)`

Fetches the list of available extended attributes for an entry.

`quint64 getExtendedAttributeSize (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl, QString attribute)`

Returns the size of value for one extended attribute for an entry.

`QByteArray getExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl, QString attribute)`

Returns the value for one extended attribute for an entry.

`void setExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl, QString attribute, QByteArray value)`

Sets the value for one extended attribute for an entry.

`void removeExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl, QString attribute)`

Removes one extended attributes for an entry.

`QMap<QString, QString> getCustomAttributes (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl)`

Returns the custom attributes for an entry.

In contrast to extended attributes, the custom attributes are not directly stored in the filesystem this way, but handle implementation specific aspects (like permissions).

`void setCustomAttribute (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl, QString attribute, QString value)`

Sets the value for one custom attribute for an entry.

See also `getCustomAttributes`.

`bool canGetFileContent (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl) = 0`

Returns if it is allowed to get the content of a certain file.

`std::shared_ptr<QIODevice> getFileContent (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl) = 0`

Get the content of a file.

`bool canCreateDirectory (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl) = 0`

Returns if it is allowed to create subdirectories in a certain directory.

`void createDirectory (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl) = 0`

Create a directory.

`bool canCreateLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl) = 0`

Returns if it is allowed to create a link in a certain directory.

`void createLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl, QString target) = 0`

Create a link.

```
bool canCreateFile (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) = 0
Returns if it is allowed to create files in a certain directory.
```

```
void createFile (sh::filesystem::Operation *op, std::shared_ptr<const
sh::filesystem::Eurl> eurl, QIODevice *content, HandlerTransfer
*handlertransfer) = 0
```

Creates a file with some content.

It may use `handlertransfer` for some better integration, like cancel support and progress monitoring.

```
bool canDeleteItem (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) = 0
Returns if it is allowed to delete a certain file/directory/link/...
```

```
void deleteItem(sh::filesystem::Operation *op, std::shared_ptr<const
               sh::filesystem::Eurl> eurl) = 0
Delete a file/directory/link/...
```

```
void deleteDirectoryIfEmpty (sh::filesystem::Operation *op, std::shared_ptr<const
                               sh::filesystem::Eurl> eurl)
    Deletes a directory only if it is empty.
```

```
bool canRenameItem (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> src) = 0
Returns if it is allowed to rename a certain file, directory, link, ... (see renameItem).
```

```
void renameItem (sh::filesystem::Operation *op, std::shared_ptr<const
sh::filesystem::Eurl> src, QString destpath) = 0
Renames an item to another path.
```

It can be a simple filename change or also changes in the deeper path segments.

```
QList<std::shared_ptr<sh::actions::AbstractActionItem>> getActions (const
                                                                    QList<std::shared_ptr<const
                                                                    sh::filesystem::Eurl>>
                                                                    eurls) = 0
```

Returns a list of actions (see former example) for showing for certain files.

Returns a list of actions (see former example) for showing for certain files.

**bool requiresResolvedLinks ()**  
Returns if this handler requires to be used by the engine with resolved links.

void **viewEnteredDirectory** (std::shared\_ptr<const *sh::filesystem::Eurl*>)  
 Callback for preparing stuff when the view entered some directory (e.g. for enabling watchers).  
 See also viewLeftDirectory.

void **viewLeftDirectory** (std::shared\_ptr<const *sh::filesystem::Eurl*>   
 Callback for preparing stuff when the view left some directory. See also viewEnteredDirectory.

bool **isWellKnownScheme** ()

Returns if the scheme for this handler is a well known one (which external programs typically would understand).

```
void search (sh::filesystem::Operation *op, std::shared_ptr<const
            sh::filesystem::Eurl> eurl, std::function<void> std::shared_ptr<const
            sh::filesystem::Eurl>, QList<std::shared_ptr<sh::search::SearchCriterion>>,
            sh::filesystem::FilesystemNodeType ftype
> addfile, std::function<void>std::shared_ptr<const sh::filesystem::Eurl>> visitdir,
QList<std::shared_ptr<sh::search::SearchCriterion>> criteria) helps searching for files.
```

Override this method with a behavior identical to the default, if a specialized implementation can be much faster than the default one.

void **customizeUi** (std::shared\_ptr<sh::filesystem::FilesystemNode> node, bool  
                   \*showSearchPanel)  
 Creates a custom gui, which becomes part of the fileview.

sh::filesystem::FilesystemModel \***model** ()

A convenience shortcut to the sh::filesystem::FilesystemModel (a singleton).

## Public Members

std::function<int (sh::filesystem::Operation \*op, std::shared\_ptr<sh::filesystem::Eurl> eurl)>  
                   **\_getType**

std::function<quint64 (sh::filesystem::Operation \*op, std::shared\_ptr<sh::filesystem::Eurl>  
                   eurl)> **\_getSize**

std::function<double (sh::filesystem::Operation \*op, std::shared\_ptr<sh::filesystem::Eurl>  
                   eurl)> **\_getMtime**

std::function<void (sh::filesystem::Operation \*op, std::shared\_ptr<sh::filesystem::Eurl> eurl,  
                   double)> **\_setMtime**

std::function<QString (sh::filesystem::Operation \*op, std::shared\_ptr<sh::filesystem::Eurl>  
                   eurl)> **\_getLinkTarget**

**std::function< QList< QString >sh::filesystem::Operation \*op, std::shared\_ptr<**

std::function<quint64 (sh::filesystem::Operation \*op, std::shared\_ptr<sh::filesystem::Eurl>  
                   eurl, QString attribute)> **\_getExtendedAttributeSize**

std::function<QByteArray (sh::filesystem::Operation \*op, std::shared\_ptr<sh::filesystem::Eurl>  
                   eurl, QString attribute)> **\_getExtendedAttribute**

std::function<void (sh::filesystem::Operation \*op, std::shared\_ptr<sh::filesystem::Eurl> eurl,  
                   QString attribute, QByteArray value)> **\_setExtendedAttribute**

std::function<void (sh::filesystem::Operation \*op, std::shared\_ptr<sh::filesystem::Eurl> eurl,  
                   QString attribute)> **\_removeExtendedAttribute**

**std::function< QList< QString >sh::filesystem::Operation \*op, std::shared\_ptr<**

std::function<void (sh::filesystem::Operation \*op, std::shared\_ptr<sh::filesystem::Eurl> eurl,  
                   QString attribute, QString value)> **\_setCustomAttribute**

std::function<QString (sh::filesystem::Operation \*op, std::shared\_ptr<sh::filesystem::Eurl>  
                   eurl)> **\_getMimeType**

std::function<bool (sh::filesystem::Operation \*op, std::shared\_ptr<sh::filesystem::Eurl>  
                   eurl)> **\_canCreateDirectory**

std::function<void (sh::filesystem::Operation \*op, std::shared\_ptr<sh::filesystem::Eurl>  
                   eurl)> **\_createDirectory**

std::function<bool (sh::filesystem::Operation \*op, std::shared\_ptr<sh::filesystem::Eurl>  
                   eurl)> **\_canCreateLink**

std::function<void (sh::filesystem::Operation \*op, std::shared\_ptr<sh::filesystem::Eurl> eurl,  
                   QString target)> **\_createLink**

std::function<bool (sh::filesystem::Operation \*op, std::shared\_ptr<sh::filesystem::Eurl>  
                   eurl)> **\_canCreateFile**

std::function<bool (sh::filesystem::Operation \*op, std::shared\_ptr<sh::filesystem::Eurl>  
                   eurl)> **\_canDeleteItem**

std::function<void (sh::filesystem::Operation \*op, std::shared\_ptr<sh::filesystem::Eurl>  
                   eurl)> **\_deleteItem**



```
std::function<bool (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> src)>
    _canRenameItem
std::function<void (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> src,
    QString destpath)> _renameItem
std::function<void (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl,
    int type, sh::filesystem::FilesystemNodeListEditor *list)> _itemlist
std::function<void (sh::filesystem::Operation *op, QList<std::shared_ptr<sh::filesystem::FilesystemNode>>)>
    _configureItems
std::function<bool (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl>
    eurl)> _canGetFileContent
std::function< std::shared_ptr< QIODevice > sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl>
    eurl, QIODevice *content, HandlerTransfer *handlertransfer)>
    _createFile
std::function< QList< std::shared_ptr< sh::actions::AbstractActionItem > > QList< sh::actions::AbstractActionItem > >
    _getItems
class ApiFilesystemOperationProgressMonitor: public sh::filesystem::FilesystemOperationProgressMonitor
#include <apifilesystemoperationprogressmonitor.h>
```

## Public Functions

**ApiFilesystemOperationProgressMonitor** (*sh::actions::ActionExecutionInfo*  
\*actionExecution)

bool **hasItemInfo** ()

Checks if this progress monitor provides information about how many items of a certain total number are transferred so far.

quint64 **doneItems** ()

Checks how many items are transferred so far.

quint64 **allItems** ()

Checks how many items are to be transferred in total (as predicted in the current moment).

bool **hasBytesInfo** ()

Checks if this progress monitor provides information about how many byte of a certain total number are transferred so far.

quint64 **doneBytes** ()

Checks how many bytes are transferred so far.

quint64 **allBytes** ()

Checks how many bytes are to be transferred in total (as predicted in the current moment).

QString **getItemInfoFrom** ()

Returns the current source of transfer (as textual information).

QString **getItemInfoTo** ()

Returns the current destination of transfer (as textual information).

QString **estimation** ()

Returns the current performance and time estimation (as textual information).



## Public Members

```
std::function<void ()> _changed
std::function<bool (QList<sh::filesystem::FilesystemOperationTransfers::OperationStep*>)>
    _resolveConflicts
class ApiGlobalObject : public sh::base::Singleton
    #include <apiglobalobject.h>
```

## Public Functions

```
std::shared_ptr<const sh::filesystem::Eurl> getEurlFromString (QString eurl)
void logDebug (QString msg)
void logInfo (QString msg)
void logWarning (QString msg)
void logError (QString msg)
int filesystemnodetype_file ()
int filesystemnodetype_directory ()
int filesystemnodetype_link ()
int filesystemnodetype_unknown ()
int filesystemnodetype_firsttype ()
int filesystemnodetype_lastphysicaltype ()
int filesystemnodetype_none ()
int filesystemnodetype_specialtreeonlydirectory ()
int filesystemnodetype_lasttype ()
int actiondefaultprecedencevalues_openfile ()
int actiondefaultprecedencevalues_builtinspecialopen ()
int messageboxbutton_ok ()
int messageboxbutton_continue ()
int messageboxbutton_cancel ()
int messageboxbutton_retry ()
int messageboxbutton_yes ()
int messageboxbutton_no ()
int displayindex_core ()
int displayindex_interesting ()
int displayindex_exotic ()
int configurationcategory_gui ()
int configurationcategory_behavior ()
int configurationcategory_externaltools ()
```

```
int settinggroup_global ()
int settinggroup_gui ()
int settinggroup_filehandling ()
int settinggroup_dataview ()
int settinggroup_behavior ()
int settinggroup_special ()
int paneldetail_positionindex_veryinteresting ()
int paneldetail_positionindex_interesting ()
int paneldetail_positionindex_exotic ()
int operationstep_conflictresolution_mergeddirectories ()
int operationstep_conflictresolution_overwritedestination ()
int operationstep_conflictresolution_renamedestinationbefore ()
int operationstep_conflictresolution_skip ()
int operationstep_conflictresolution_unresolved ()
int operationstep_conflictresolution_indirect ()
int operationstep_conflictresolution_usedifferentdestinationname ()
int filepropertydialogtab_index_core ()
int filepropertydialogtab_index_veryimportant ()
int filepropertydialogtab_index_important ()
int filepropertydialogtab_index_normal ()
int filepropertydialogtab_index_exotic ()
int filepropertydialogtab_propertytype_string ()
int filepropertydialogtab_propertytype_stringmap ()
int filepropertydialogtab_propertytype_icontextbanner ()
int searchcriterionfactory_index_core ()
int searchcriterionfactory_index_normal ()
int searchcriterionfactory_index_exotic ()
int thumbnailprovider_index_core ()
int thumbnailprovider_index_normal ()
int thumbnailprovider_index_exotic ()
int thumbnailprovider_index_fallback ()
void register_predicatedactionfactory (std::shared_ptr<sh::scripting::api::ApiActionFactory>
                                       f)
bool isDebugBuild ()
std::shared_ptr<sh::scripting::api::ApiFilesystemHandler> create_FilesystemHandler ()
```

```

std::shared_ptr<sh::scripting::api::ApiActionActionItem> create_ActionActionItem (QString
                                                                    text,
                                                                    bool
                                                                    en-
                                                                    abled,
                                                                    QString
                                                                    icon,
                                                                    int
                                                                    de-
                                                                    fault-
                                                                    Ac-
                                                                    tion-
                                                                    Prece-
                                                                    dence)

std::shared_ptr<sh::scripting::api::ApiSubmenuItem> create_SubmenuItem (QString
                                                                    text,
                                                                    bool
                                                                    en-
                                                                    abled,
                                                                    QString
                                                                    icon,
                                                                    int
                                                                    de-
                                                                    fault-
                                                                    Ac-
                                                                    tion-
                                                                    Prece-
                                                                    dence)

std::shared_ptr<sh::scripting::api::ApiDetailColumn> create_DetailColumn (QString
                                                                    name,
                                                                    QString
                                                                    display-
                                                                    Name,
                                                                    int
                                                                    dis-
                                                                    playIn-
                                                                    dex, bool
                                                                    sort_doTypediff,
                                                                    int
                                                                    de-
                                                                    fault-
                                                                    Width,
                                                                    bool
                                                                    is-
                                                                    RightAl-
                                                                    igned)

std::shared_ptr<sh::scripting::api::ApiFilePropertyDialogTab> create_FilePropertyDialogTab (QString
                                                                    ti-
                                                                    tle,
                                                                    QList<
                                                                    prop-
                                                                    er-
                                                                    ties)

std::shared_ptr<sh::scripting::api::ApiFilePropertyDialogTabFactory> create_FilePropertyDialogTabFactory

```

```
std::shared_ptr<sh::scripting::api::ApiActionFactory> create_ActionFactory (QString
                                                    cate-
                                                    gory,
                                                    QList<std::shared_ptr<action
                                                    predi-
                                                    cates)

std::shared_ptr<sh::scripting::api::ApiPanelDetailFactorySingle> create_DetailFactorySingle (int
                                                                 po-
                                                                 si-
                                                                 tion,
                                                                 int
                                                                 val-
                                                                 ueWidth
                                                                 Hint)

std::shared_ptr<sh::scripting::api::ApiPanelDetailFactoryMulti> create_DetailFactoryMulti (int
                                                                 po-
                                                                 si-
                                                                 tion,
                                                                 int
                                                                 val-
                                                                 ueWidth
                                                                 Hint)

std::shared_ptr<sh::scripting::api::ApiFileSystemOperationProgressMonitor> create_FileSystemOperation

std::shared_ptr<sh::filesystem::FilesystemNode> _createFilesystemNode (std::shared_ptr<sh::filesystem::FilesystemNode> parent,
                                                                    eurl,
                                                                    sh::scripting::api::ApiFileSystemOperationProgressMonitor *handler,
                                                                    int
                                                                    nodetype,
                                                                    std::shared_ptr<sh::filesystem::FilesystemNode> parent, bool
                                                                    doInitialLoad, bool
                                                                    showInitialLoadLabel,
                                                                    bool isHidden)

std::shared_ptr<sh::scripting::api::ApiThumbnailProvider> create_ThumbnailProvider ()
```

```

std::shared_ptr<sh::filesystem::FilesystemNode> _getOrCreateFilesystemNode (std::shared_ptr<sh::filesystem::Eurl> eurl,
sh::scripting::api::ApiFilesystemHandler *handler,
int node-type,
std::shared_ptr<sh::filesystem::FilesystemNode> parent,
bool doInitialLoading,
bool showInitialLoadingLabel,
bool isHidden)

void _register_FilesystemHandler (std::shared_ptr<sh::scripting::api::ApiFilesystemHandler> handler, QString scheme)

std::shared_ptr<sh::filesystem::FilesystemNode> modelRootNode ()

QList<std::shared_ptr<sh::filesystem::FilesystemNode>> _findFilesystemNodesForEurl (std::shared_ptr<sh::filesystem::Eurl> eurl)

QList<std::shared_ptr<sh::filesystem::FilesystemNode>> _tryGetFilesystemNodesForEurl (std::shared_ptr<sh::filesystem::Eurl> eurl)

void refreshData (std::shared_ptr<const sh::filesystem::Eurl> eurl, bool forceFindParent, bool withDetails)

std::shared_ptr<sh::filesystem::Eurl> createEurl (QString scheme, QString hostname, QString path)

std::shared_ptr<sh::filesystem::Operation> createOperation ()

sh::scripting::StringMap<QString> createArgumentException ()

sh::scripting::StringMap<QString> createIOException ()

sh::scripting::StringMap<QString> createRuntimeException ()

sh::scripting::StringMap<QString> createProgramException ()

sh::scripting::StringMap<QString> createException ()

std::shared_ptr<sh::scripting::api::ApiReadDataDevice> _createReadDataDevice ()

std::shared_ptr<ApiSetting> createSetting (QString name, QString description, int group, bool isAdvancedSetting, bool isGlobal, bool isPerFileview)

std::shared_ptr<sh::scripting::api::ApiThread> create_Thread ()

std::shared_ptr<sh::scripting::api::ApiTimer> create_Timer ()

```

```
std::shared_ptr<sh::scripting::api::ApiSearchCriterion> create_SearchCriterion (std::shared_ptr<sh::scrip-
                                                    fac-
                                                    tory)

std::shared_ptr<sh::scripting::api::ApiSearchCriterionFactory> create_SearchCriterionFactory (QStrin-
                                                    key,
                                                    QStrin-
                                                    de-
                                                    scrip-
                                                    tion)

sh::ui::MainWindow *mainwindow ()

void registerPanelDetailFactorySingle (std::shared_ptr<sh::scripting::api::ApiPanelDetailFactorySin-
void registerPanelDetailFactoryMulti (std::shared_ptr<sh::scripting::api::ApiPanelDetailFactoryMul-
void registerSetting (std::shared_ptr<sh::settings::Setting> setting)
void openItems (QList<std::shared_ptr<const sh::filesystem::Eurl>> eurls)
void registerFilePropertyDialogTabFactory (int index,
std::shared_ptr<sh::scripting::api::ApiFilePropertyDia-
void registerThumbnailProvider (int idx, std::shared_ptr<sh::scripting::api::ApiThumbnailProvider>
thumbnailprovider)
void registerSearchCriterionFactory (int idx, std::shared_ptr<ApiSearchCriterionFactory>
apicrit)
std::shared_ptr<sh::configuration::ConfigurationValue> register_ConfigurationValueInt (QString
name,
int
de-
flt,
QString
desc,
int
cat-
e-
gory,
QString
longdesc,
QString
change-
hint)
```

```
std::shared_ptr<sh::configuration::ConfigurationValue> register_ConfigurationValueFloat (QString
                                                    name,
                                                    int
                                                    de-
                                                    flt,
                                                    QString
                                                    desc,
                                                    int
                                                    cat-
                                                    e-
                                                    gory,
                                                    QString
                                                    longdesc,
                                                    QString
                                                    change-
                                                    hint)

std::shared_ptr<sh::configuration::ConfigurationValue> register_ConfigurationValueBool (QString
                                                    name,
                                                    bool
                                                    de-
                                                    flt,
                                                    QString
                                                    desc,
                                                    int
                                                    cat-
                                                    e-
                                                    gory,
                                                    QString
                                                    longdesc,
                                                    QString
                                                    change-
                                                    hint)

std::shared_ptr<sh::configuration::ConfigurationValue> register_ConfigurationValueString (QString
                                                    name,
                                                    QString
                                                    de-
                                                    flt,
                                                    QString
                                                    desc,
                                                    int
                                                    cat-
                                                    e-
                                                    gory,
                                                    QString
                                                    longdes
                                                    QString
                                                    change-
                                                    hint)

void registerTransferrableDetailColumn (int i,
                                         std::shared_ptr<sh::filesystem::DetailColumn>
                                         detailColumn)

std::shared_ptr<sh::filesystem::DetailColumn> findDetailColumnByName (QString
                                                                    name)
```

```
QString shallotDataDir ()
QString shallotBuiltinPluginDir ()
QString shallotSystemPluginDir ()
QString shallotUserPluginDir ()
QString shallotLanguage ()
sh::tools::BookmarkManager *bookmarkManager ()
std::shared_ptr<sh::actions::Predicate> create_OnDirectoriesPredicate ()
std::shared_ptr<sh::actions::Predicate> create_OnFilesPredicate ()
std::shared_ptr<sh::actions::Predicate> create_OnLinksPredicate ()
std::shared_ptr<sh::actions::Predicate> create_OnSingleEntrySelectionPredicate ()
std::shared_ptr<sh::actions::Predicate> create_DontResolveLinksPredicate ()
std::shared_ptr<sh::actions::Predicate> create_HideOnCurrentDirectoryLevelPredicate ()
std::shared_ptr<sh::actions::Predicate> create_HideOnSelectionLevelPredicate ()
std::shared_ptr<sh::actions::Predicate> create_ByRegExpPredicate (QString    pat-
                                                                tern)
std::shared_ptr<sh::actions::Predicate> create_KeyShortcutPredicate (QString
                                                                    shortcut,
                                                                    bool   trig-
                                                                    gersOnCur-
                                                                    rentDirec-
                                                                    toryLevel)
std::shared_ptr<sh::actions::Predicate> create_PositionIndexPredicate (int
                                                                    index)

void doInitialize ()
    Executes singleton initialization.

void doShutdown ()
    Executes singleton shutdown.

void shutdown ()
    Shutdown down this singleton.

bool isAlive ()
    Returns if this singleton is alive (true until its shutdown begins).
```

## Private Functions

```
ApiGlobalObject ()

class ApiHelperMethods
    #include <apihelpermethods.h>
```



## Public Static Functions

```

QByteArray QIODevice_read (QIODevice *self)
QByteArray QIODevice_readall (QIODevice *self)
QByteArray ApiReadDataDevice_read (sh::scripting::api::ApiReadDataDevice *self)
QByteArray ApiReadDataDevice_readall (sh::scripting::api::ApiReadDataDevice
                                         *self)
void FilesystemNodeList_resetItems (sh::filesystem::FilesystemNodeList
                                     *self, int type,
                                     QList<std::shared_ptr<sh::filesystem::FilesystemNode>>
                                     list)
void FilesystemNode_addDetail (filesystem::FilesystemNode *self,
                               std::shared_ptr<sh::filesystem::DetailColumn>
                               col)
void FilesystemNode_setIcon (sh::filesystem::FilesystemNode *self, QString icon)
void FilesystemNode_setDisplayName (filesystem::FilesystemNode *self, QString
                                     displayname)
bool FilesystemNode_isHidden (filesystem::FilesystemNode *self)
void FilesystemNode_setHidden (filesystem::FilesystemNode *self, bool v)
void ConfigurationValue_setConfigValueInt (sh::configuration::ConfigurationValue
                                             *self, int v)
void ConfigurationValue_setConfigValueFloat (sh::configuration::ConfigurationValue
                                              *self, double v)
void ConfigurationValue_setConfigValueBool (sh::configuration::ConfigurationValue
                                              *self, bool v)
void ConfigurationValue_setConfigValueString (sh::configuration::ConfigurationValue
                                              *self, QString v)
QList<std::shared_ptr<sh::filesystem::Eurl>> FilesystemOperation_itemlist (sh::filesystem::FilesystemOperation
                                                                            *self,
                                                                            std::shared_ptr<const
                                                                            sh::filesystem::Eurl>
                                                                            eurl)
QList<std::shared_ptr<sh::filesystem::Eurl>> FilesystemOperation_itemlistByType (sh::filesystem::FilesystemOperation
                                                                                   *self,
                                                                                   std::shared_ptr<const
                                                                                   sh::filesystem::Eurl>
                                                                                   eurl,
                                                                                   int
                                                                                   ntype)
int FilesystemOperation_getType (sh::filesystem::FilesystemOperation *self,
                                  std::shared_ptr<const sh::filesystem::Eurl>
                                  eurl)
void FilesystemOperation_createFile (sh::filesystem::FilesystemOperation
                                       *self, std::shared_ptr<const
                                       sh::filesystem::Eurl> eurl,
                                       sh::scripting::api::ApiReadDataDevice
                                       *content, std::shared_ptr<filesystem::FilesystemOperationProgressmon>
                                       progressmon)

```

```
QList<std::shared_ptr<sh::filesystem::FileSystemNode>> FileSystemOperation_resolveNodeLink (sh::filesystem::FileSystemNode *self,
                                                                                          std::shared_ptr<const sh::filesystem::Eurl> eurl)
noa

QList<std::shared_ptr<sh::filesystem::FileSystemNode>> FileSystemOperation_resolveNodeLink_NonRecursive (sh::filesystem::FileSystemNode *self,
                                                                                          std::shared_ptr<const sh::filesystem::Eurl> eurl)

std::shared_ptr<sh::filesystem::Eurl> FileSystemOperation_resolveEurlLink (sh::filesystem::FileSystemNode *self,
                                                                                          std::shared_ptr<const sh::filesystem::Eurl> eurl)

std::shared_ptr<sh::filesystem::Eurl> FileSystemOperation_resolveEurlLink_NonRecursive (sh::filesystem::FileSystemNode *self,
                                                                                          std::shared_ptr<const sh::filesystem::Eurl> eurl)

void PanelDetail_setRow (sh::paneldetails::PanelDetail *self,   QString   key,
                        QList<QString> value)

void MainWindow_jumpToEurl (sh::ui::MainWindow *self,   std::shared_ptr<const sh::filesystem::Eurl> eurl)

std::shared_ptr<sh::filesystem::FileSystemNode> MainWindow_getCurrentDirectoryNode (sh::ui::MainWindow *self)

void ApiActionActionItem_setEnabled (sh::scripting::api::ApiActionActionItem *self, bool enabled)

bool ApiActionActionItem_enabled (sh::scripting::api::ApiActionActionItem *self)

void ApiActionActionItem_setVisible (sh::scripting::api::ApiActionActionItem *self, bool visible)

bool ApiActionActionItem_visible (sh::scripting::api::ApiActionActionItem *self)

void ApiSubmenuItem_setSubitems (sh::scripting::api::ApiSubmenuItem *self, QList<std::shared_ptr<actions::AbstractActionItem>> subitems)

void ApiSubmenuItem_setEnabled (sh::scripting::api::ApiSubmenuItem *self, bool enabled)

bool ApiSubmenuItem_enabled (sh::scripting::api::ApiSubmenuItem *self)

void ApiSubmenuItem_setVisible (sh::scripting::api::ApiSubmenuItem *self, bool visible)

bool ApiSubmenuItem_visible (sh::scripting::api::ApiSubmenuItem *self)

int OperationStep_conflictResolution (filesystem::FileSystemOperationTransfers::OperationStep *self)

bool FileSystemOperationProgressMonitor_hasItemInfo (sh::scripting::api::ApiFileSystemOperationProgressMonitor *self)

quint64 FileSystemOperationProgressMonitor_doneItems (sh::scripting::api::ApiFileSystemOperationProgressMonitor *self)
```

```

quint64 FilesystemOperationProgressMonitor_allItems (sh::scripting::api::ApiFilesystemOperationProgressMonitor
                                                    *self)

bool FilesystemOperationProgressMonitor_hasBytesInfo (sh::scripting::api::ApiFilesystemOperationProgressMonitor
                                                    *self)

quint64 FilesystemOperationProgressMonitor_doneBytes (sh::scripting::api::ApiFilesystemOperationProgressMonitor
                                                    *self)

quint64 FilesystemOperationProgressMonitor_allBytes (sh::scripting::api::ApiFilesystemOperationProgressMonitor
                                                    *self)

QString FilesystemOperationProgressMonitor_getItemInfoFrom (sh::scripting::api::ApiFilesystemOperationProgressMonitor
                                                            *self)

QString FilesystemOperationProgressMonitor_getItemInfoTo (sh::scripting::api::ApiFilesystemOperationProgressMonitor
                                                            *self)

QString FilesystemOperationProgressMonitor_estimation (sh::scripting::api::ApiFilesystemOperationProgressMonitor
                                                        *self)

QList<std::shared_ptr<sh::filesystem::FilesystemNode>> ApiFilePropertyDialogTab_nodes (sh::scripting::api::ApiFilePropertyDialogTab
                                                                                      *self)

int ActionExecutionUserFeedback_messageBox (sh::actions::ActionExecutionUserFeedback
                                             *self,      QString      m,
                                             QList<QString> l,   QString
s, int i, int j)

QString ActionExecutionUserFeedback_simpleInputDialog (sh::actions::ActionExecutionUserFeedback
                                                         *self, QString text,
                                                         QString deflt)

int FilePropertyDialogTab_selectedIndexForProperty (ApiFilePropertyDialogTab
                                                    *self, quintptr wid-
                                                    get)

void FilePropertyDialogTab_refresh (ApiFilePropertyDialogTab *self)

class ApiPanelDetailFactoryMulti : public sh::paneldetails::PanelDetailFactoryForFunctionMulti
#include <apipaneldetailfactory.h>

```

## Public Functions

**ApiPanelDetailFactoryMulti** (int position, int valueWidthHint)

```

void setvalue (std::shared_ptr<sh::paneldetails::PanelDetail> detail,
               QList<std::shared_ptr<sh::filesystem::FilesystemNode>> nodes, filesystem::Operation *op)

```

```

void linktriggered (QList<std::shared_ptr<sh::filesystem::FilesystemNode>> nodes,
                    QString linktarget)

```

```

std::shared_ptr<PanelDetail> construct (QList<std::shared_ptr<sh::filesystem::FilesystemNode>>
                                         nodes, sh::filesystem::Operation *op)

```

### Public Members

```
std::function<void (sh::paneldetails::PanelDetail*, QList<std::shared_ptr<sh::filesystem::FilesystemNode>>,
    sh::filesystem::Operation *op)> _setvalue
std::function<void (QList<std::shared_ptr<sh::filesystem::FilesystemNode>>,      QString)>
    _linktriggered
class ApiPanelDetailFactorySingle : public sh::paneldetails::PanelDetailFactoryForFunctionSingle
#include <apipaneldetailfactory.h>
```

### Public Functions

```
ApiPanelDetailFactorySingle (int position, int valueWidthHint)
void setvalue (std::shared_ptr<sh::paneldetails::PanelDetail>,
    std::shared_ptr<sh::filesystem::FilesystemNode> node,      filesystem::
    Operation *op)
void linktriggered (std::shared_ptr<sh::filesystem::FilesystemNode> node, QString link-
    target)
std::shared_ptr<PanelDetail> construct (QList<std::shared_ptr<sh::filesystem::FilesystemNode>>
    nodes, sh::filesystem::Operation *op)
```

### Public Members

```
std::function<void (sh::paneldetails::PanelDetail*, std::shared_ptr<sh::filesystem::FilesystemNode>,
    sh::filesystem::Operation *op)> _setvalue
std::function<void (std::shared_ptr<sh::filesystem::FilesystemNode>,      QString)>
    _linktriggered
class ApiReadDataDevice : public sh::tools::ReadDataDevice
#include <apireaddatadevice.h>
```

### Public Functions

```
ApiReadDataDevice ()
QByteArray getdata ()
    This method returns the next available chunk of data. It may return empty ar-
    rays whenever temporarily no data is available. Returning a null array (with
    QByteArray::isNull() == true) marks the end of the stream.
void _ended ()
```

### Public Members

```
std::function<QByteArray ()> _getdata
```

## Private Members

bool **isended**

```
class ApiSearchCriterion : public sh::search::criteria::AbstractActionDrivenSearchCriterion
    #include <apisearchcriterion.h>
```

## Public Functions

**ApiSearchCriterion** (std::shared\_ptr<sh::search::SearchCriterionFactory> *factory*)

bool **match2** (sh::filesystem::Operation \**op*, std::shared\_ptr<const sh::filesystem::Eurl> *eurl*)

QList<QString> **getsearchspec** ()

bool **match** (sh::filesystem::Operation \**op*, std::shared\_ptr<const sh::filesystem::Eurl> *eurl*)

Checks if a given file (by eurl) matches this criterion. .

QString **valuedescription** ()

Returns a human readable text describing the current criterion shortly and precisely. .

QString **serialize** ()

Serializes this criterion to a string. See also *SearchCriterion::deserialize*.

std::shared\_ptr<sh::search::SearchCriterionFactory> **factory** ()

Returns the factory this criterion has created (provides some more metadata).

## Public Members

std::function<void (sh::actions::ActionExecutionInfo\*)> **\_configure**

std::function<bool (sh::filesystem::Operation \**op*, std::shared\_ptr<sh::filesystem::Eurl> *eurl*)> **\_match**

QStringList **searchspec**

## Public Static Functions

void **\_createEditor** (std::shared\_ptr<sh::search::SearchCriterion> *cfg*,  
sh::ui::SearchPanelConfiguration \**panelcfg*)

void **\_editorUpdateConfiguration** (sh::ui::SearchPanelConfiguration \**panelcfg*,  
std::shared\_ptr<sh::search::SearchCriterion> *c*)

std::shared\_ptr<sh::search::SearchCriterion> **deserialize** (QString *s*)

Deserializes this criterion from a string. See also *SearchCriterion::serialize*.

```
class ApiSearchCriterionFactory : public sh::search::SearchCriterionFactory
    #include <apisearchcriterion.h>
```

## Public Functions

**ApiSearchCriterionFactory** (QString *key*, QString *description*)

QString **key** ()

Returns the string key for the associated search criterion class. .

QString **description** ()

Returns the description string for the associated search criterion class. .

void **editor** (std::shared\_ptr<sh::search::SearchCriterion> *c*,  
sh::ui::SearchPanelConfiguration \*panelcfg)

void **editorupdateconfig** (sh::ui::SearchPanelConfiguration \*panelcfg,  
std::shared\_ptr<sh::search::SearchCriterion> *c*)

std::shared\_ptr<sh::search::SearchCriterion> **construct** ()

Constructs an instance of the associated search criterion class. .

bool **isVisibleFor** (sh::filesystem::Operation \*op, std::shared\_ptr<sh::filesystem::FileSystemNode>  
*node*)

Checks if the associated search criterion class should be offered to the user for a given node. .

void **editor** (std::shared\_ptr<sh::search::SearchCriterion> *c*,  
sh::ui::SearchPanelConfiguration \*panelcfg) = 0

Builds the search config ui in a search panel.

Takes the data from *c* and populates *panelcfg* with it.

void **editorupdateconfig** (sh::ui::SearchPanelConfiguration \*panelcfg,  
std::shared\_ptr<sh::search::SearchCriterion> *c*) = 0

Updates own data with the content from the search config ui in a search panel.

Takes the data from *panelcfg* and updates *c* with it.

## Public Members

std::function<QString (QList<QString>)> **\_getsearchspecdesc**

std::function< std::shared\_ptr< sh::search::SearchCriterion >> **\_construct**

std::function<bool (sh::filesystem::Operation\*, std::shared\_ptr<sh::filesystem::FileSystemNode>)>  
**\_isVisibleFor**

**class ApiSetting**: public sh::settings::Setting

#include <apisetting.h>

## Public Functions

**ApiSetting** (QString *name*, QString *description*, sh::settings::SettingGroup *group*, bool  
*isAdvancedSetting*, bool *isGlobal*, bool *isPerFileview*)

QString **name** ()

Gets the internal name.

QString **description** ()

Gets the description text.

sh::settings::SettingGroup **group** ()

Gets the group;.

```

bool isAdvancedSetting ()
    Is this an advanced setting?

void setValue (QString value)
    Called from Shallot core when the value was set (for a not-per-fileview setting).

void setValue (sh::ui::FileView *filelist, QString value)
    Called from Shallot core when the value was set (for a per-fileview setting).

QString getValue (sh::ui::FileView *filelist)
    Get the currently set value.

bool isGlobal ()
    Does this setting apply globally or just for a certain subtree of nodes?

bool isPerFileview ()
    Does this setting apply for each fileview individually or for the complete main window?

QString valueDescription (QString value)
    Gets a human readable description text for a value.

```

### Public Members

```

std::function<void (QString)> _setValue1
std::function<void (int, QString)> _setValue2
std::function<QString (int)> _getValue
std::function<QString (QString)> _valueDescription
QString _name
QString _description
sh::settings::SettingGroup _group
bool _isAdvancedSetting
bool _isGlobal
bool _isPerFileview

```

### Public Static Functions

```

QString getGroupDescription (int g)
    Low-level function which gets the description text of a group.

class ApiSubmenuItemActionItem: public sh::actions::SubmenuItemActionItem
    #include <apiaction.h>

```

## Public Functions

**ApiSubmenuItem**(QString *text*, bool *enabled*, QString *icon*, int *defaultActionPrecedence*)

void **initialize**() **override**

Initialize the action. This should make the time-consuming parts, e.g. for determining a label or enabled state.

**const** QList<std::shared\_ptr<[sh::actions::AbstractActionItem](#)>> **subitems**()

Returns the list of subitems from this submenu.

void **setSubitems**(**const**       QList<std::shared\_ptr<[sh::actions::AbstractActionItem](#)>>  
                                  *subitems*)

Sets the subitems.

QString **text**()

Returns the displayed text for this action.

QString **icon**()

Returns the icon for this action.

bool **enabled**()

Checks if this action is enabled.

bool **isChecked**()

Checks if this action is checked (has a cross in the ui).

bool **isCheckable**()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence**() **const**

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText**(QString *text*)

Sets the displayed text.

void **setIcon**(QString *icon*)

Sets the icon.

void **setEnabled**(bool *enabled*)

Sets if the item is enabled.

void **setChecked**(bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible**(bool *visible*)

Sets the visibility of this item.

bool **visible**()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<AbstractActionItem> **parentAction**()

Returns the parent action, if it is added to a container.

void **\_setParentAction**(std::shared\_ptr<AbstractActionItem> *parent*)

Sets the parent action. .



```

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.

```

## Public Members

```
std::function<void ()> _initialize
```

## Signals

```

void subitemsChanged ()
    Emits when the list of subitems changed.

void changed ()
    Emits when some data changed.

```

```

class ApiThread: public std::enable_shared_from_this<ApiThread>
    #include <apithread.h>

```

## Public Functions

```

ApiThread ()

void start ()

```

## Public Members

```
std::function<void ()> _run
```

```

class ApiThumbnailProvider: public sh::tools::ThumbnailProvider
    #include <apithumbnailprovider.h>

```

## Public Functions

```

ApiThumbnailProvider ()

void getThumbnail (sh::filesystem::Operation                *operation,
                  std::shared_ptr<sh::filesystem::FileSystemNode> node,    QString
                  contentType, int width, int height, QIcon *icon)

```

## Public Members

```
std::function<QByteArray (sh::filesystem::Operation*, std::shared_ptr<sh::filesystem::FileSystemNode>,
                          QString, int, int) > _getThumbnail

class ApiTimer : public std::enable_shared_from_this<ApiTimer>
#include <apitimer.h>
```

## Public Functions

```
ApiTimer ()
void start (int interval)
void stop ()
```

## Public Members

```
std::function<void () > _run
```

## Private Members

```
QMutex _mutex
QTimer _timer
bool _isexecuting = false
QList<std::shared_ptr<ApiTimer>> _runningTimers

namespace pythoninterop
Interoperation layer between the Shallot core (C++) and Python plugins.
```

## Typedefs

```
using StringMap = QMap<QString, V>
```

## Functions

```
PyObject *getPyNone ()
bool isPyNone (PyObject *o)
void *getPointerFromShallotPyObject (PyObject *o)
std::shared_ptr<void> getSharedPointerFromShallotPyObject (PyObject *o)
PyObject *getShallotPyObjectFromPointer (void *o, sh::scripting::ApiClass *api-
                                          class)
PyObject *getShallotPyObjectFromSharedPointer (std::shared_ptr<void> o,
                                                sh::scripting::ApiClass *api-
                                                class)
sh::scripting::ApiClass *getApiClass (QString typeIdName)
PyObject *PyTupleGetItem (PyObject *o, int itm)
PyObject *PyListGetItem (PyObject *o, int itm)
```

```

int PyTupleCount (PyObject *o)
int PyListCount (PyObject *o)
bool PyIsTuple (PyObject *o)
bool PyIsList (PyObject *o)
bool PyIsUnicode (PyObject *o)
bool PyIsBytes (PyObject *o)
bool PyIsDict (PyObject *o)
bool PyIsBool (PyObject *o)
bool PyIsFloat (PyObject *o)
bool PyIsLong (PyObject *o)
void PyIncRef (PyObject *o)
void PyDecRef (PyObject *o)
void PyTupleSetItem (PyObject *o, int itm, PyObject *val)
PyObject *PyObjectCall (PyObject *fct, PyObject *args)
PyObject *PyTupleNew (int cnt)
PyObject *PyDictNew ()
PyObject *PyDictKeys (PyObject *o)
PyObject *PyDictGetItem (PyObject *dict, PyObject *key)
void PyDictSetItem (PyObject *dict, PyObject *key, PyObject *val)
bool PythonToC_ExecuteGuarded (std::function<void>
    > fct)
void invokePython (std::function<void>
    > fct)

template<typename R, typename T, typename ...ArgsF, typename ...ArgsT>
R CallNativeFunctionWithTuple (T *pObj, R (T::* f)) ArgsF...
    , std::tuple<ArgsT...> const &t

template<typename R, typename T, typename ...ArgsF, typename ...ArgsT>
R CallNativeHelperFunctionWithTuple (T *pObj, R (*f)) T*, ArgsF...
    , std::tuple<ArgsT...> const &t

template<int pos, class ...Args>
class _PyObjectFunctionToNativeFunction_SetTupleValue
    #include <pythonscriptinterpreter.h>

template<int pos>
class _PyObjectFunctionToNativeFunction_SetTupleValue<pos>
    #include <pythonscriptinterpreter.h>

```

### Public Static Functions

```
void setTupleValue (PyObject*)

template<int pos, class Arg, class ...Args>
class _PyObjectFunctionToNativeFunction_SetTupleValue<pos, Arg, Args...>
    #include <pythonscriptinterpreter.h>
```

### Public Static Functions

```
void setTupleValue (PyObject *tup, Arg arg, Args... args)

template<int pos, class ...Args>
class _PyObjectFunctionToNativeFunction_TupleValueSetter
    #include <pythonscriptinterpreter.h> This class is a workaround for gcc<4.9 bug 55914. It was
    not able to call ...::setTupleValue directly from within the invokePython lambda
```

### Public Functions

```
_PyObjectFunctionToNativeFunction_TupleValueSetter (Args... args)

void set (PyObject *tuple)
```

### Private Members

```
std::function<void (PyObject*)> _set
```

### Private Static Functions

```
void __set (Args... args, PyObject *tuple)

template<typename T, T>
struct CallNativeHelperMethodWithPyObjectArguments
    #include <pythonscriptinterpreter.h>

template<typename T, typename R, typename ... Args, R(*)(T *, Args...) MF> CallNativeHelperMethod
    #include <pythonscriptinterpreter.h>
```

### Public Static Functions

```
PyObject *call (PyObject *a, PyObject *b)

template<typename T, typename ... Args, void(*)(T *, Args...) MF> CallNativeHelperMethod
    #include <pythonscriptinterpreter.h>
```

### Public Static Functions

```

PyObject *call (PyObject *a, PyObject *b)

template<typename T, T>
struct CallNativeMethodWithPyObjectArguments
    #include <pythonscriptinterpreter.h>

template<typename T, typename R, typename ... Args, R(T::*)(Args...) const MF> *
    #include <pythonscriptinterpreter.h>

```

### Public Static Functions

```

PyObject *call (PyObject *a, PyObject *b)

template<typename T, typename R, typename ... Args, R(T::*)(Args...) MF> * (Args...)
    #include <pythonscriptinterpreter.h>

```

### Public Static Functions

```

PyObject *call (PyObject *a, PyObject *b)

template<typename T, typename ... Args, void(T::*)(Args...) const MF> * (Args...)
    #include <pythonscriptinterpreter.h>

```

### Public Static Functions

```

PyObject *call (PyObject *a, PyObject *b)

template<typename T, typename ... Args, void(T::*)(Args...) MF> * (Args...), MF>
    #include <pythonscriptinterpreter.h>

```

### Public Static Functions

```

PyObject *call (PyObject *a, PyObject *b)

class Converters
    #include <pythonscriptinterpreter.h>

```

### Public Static Functions

```

QString getStringFromPyObject (PyObject *o)
QVariant getVariantFromPyObject (PyObject *o)
int getIntFromPyObject (PyObject *o)
double getDoubleFromPyObject (PyObject *o)
qint64 getInt64FromPyObject (PyObject *o)
quint64 getUInt64FromPyObject (PyObject *o)
bool getBoolFromPyObject (PyObject *o)
QByteArray getByteArrayFromPyObject (PyObject *o)

```

```
void *getPointerFromPyObject (PyObject *o)
std::shared_ptr<void> getSharedPointerFromPyObject (PyObject *o)
PyObject *getPyObjectFromString (QString v)
PyObject *getPyObjectFromVariant (QVariant v)
PyObject *getPyObjectFromInt (int v)
PyObject *getPyObjectFromDouble (double v)
PyObject *getPyObjectFromInt64 (qint64 v)
PyObject *getPyObjectFromUInt64 (quint64 v)
PyObject *getPyObjectFromBool (bool v)
PyObject *getPyObjectFromByteArray (QByteArray v)

template<int mode, uint N>
struct FunctionBindTuple
    #include <pythonscriptinterpreter.h>

template<>
struct FunctionBindTuple<0, 0>
    #include <pythonscriptinterpreter.h>
```

### Public Static Functions

```
template<typename R, typename T, typename ...ArgsF, typename ...ArgsT, typename ...Args>
R applyTuple (T *pObj, R (T::*f)) ArgsF...
    , const std::tuple<ArgsT...>&, Args... args

template<uint N>
struct FunctionBindTuple<0, N>
    #include <pythonscriptinterpreter.h>
```

### Public Static Functions

```
template<typename R, typename T, typename ...ArgsF, typename ...ArgsT, typename ...Args>
R applyTuple (T *pObj, R (T::*f)) ArgsF...
    , const std::tuple<ArgsT...> &t, Args... args

template<>
struct FunctionBindTuple<1, 0>
    #include <pythonscriptinterpreter.h>
```

### Public Static Functions

```
template<typename R, typename T, typename ...ArgsF, typename ...ArgsT, typename ...Args>
R applyTuple (T *pObj, R (*f)) T*, ArgsF...
    , const std::tuple<ArgsT...>&, Args... args

template<uint N>
struct FunctionBindTuple<1, N>
    #include <pythonscriptinterpreter.h>
```

### Public Static Functions

```

template<typename R, typename T, typename ...ArgsF, typename ...ArgsT, typename ...Args>
R applyTuple (T *pObj, R (*f)) T*, ArgsF...
, const std::tuple<ArgsT...> &t, Args... args

template<class V>
class NativeToPyObject
    #include <pythonscriptinterpreter.h> Helps to return a Python object from a native c++ value
    (with reference ownership)

template<>
class NativeToPyObject<bool>
    #include <pythonscriptinterpreter.h>

```

### Public Static Functions

```

PyObject *toPyObject (bool v)

template<>
class NativeToPyObject<double>
    #include <pythonscriptinterpreter.h>

```

### Public Static Functions

```

PyObject *toPyObject (double v)

template<>
class NativeToPyObject<int>
    #include <pythonscriptinterpreter.h>

```

### Public Static Functions

```

PyObject *toPyObject (int v)

template<>
class NativeToPyObject<QByteArray>
    #include <pythonscriptinterpreter.h>

```

### Public Static Functions

```

PyObject *toPyObject (QByteArray v)

template<>
class NativeToPyObject<qint64>
    #include <pythonscriptinterpreter.h>

```

### Public Static Functions

```
PyObject *toPyObject (qint64 v)

template<typename V>
class NativeToPyObject<QList<V>>
    #include <pythonscriptinterpreter.h>
```

### Public Static Functions

```
PyObject *toPyObject (QList<V> v)

template<typename V>
class NativeToPyObject<QMap<QString, V>>
    #include <pythonscriptinterpreter.h>
```

### Public Static Functions

```
PyObject *toPyObject (QMap<QString, V> v)

template<>
class NativeToPyObject<QString>
    #include <pythonscriptinterpreter.h>
```

### Public Static Functions

```
PyObject *toPyObject (QString v)

template<>
class NativeToPyObject<qint64>
    #include <pythonscriptinterpreter.h>
```

### Public Static Functions

```
PyObject *toPyObject (qint64 v)

template<>
class NativeToPyObject<QVariant>
    #include <pythonscriptinterpreter.h>
```

### Public Static Functions

```
PyObject *toPyObject (QVariant v)

template<class V> shared_ptr< const V > >
    #include <pythonscriptinterpreter.h>
```



### Public Static Functions

```

PyObject *toPyObject (std::shared_ptr<const V> v)
template<class V> shared_ptr< V > >
#include <pythonscriptinterpreter.h>

```

### Public Static Functions

```

PyObject *toPyObject (std::shared_ptr<V> v)
template<class V>
class NativeToPyObject<V*>
#include <pythonscriptinterpreter.h>

```

### Public Static Functions

```

PyObject *toPyObject (V *v)
template<class R, class ...Args>
class PyObjectFunctionToNativeFunction
#include <pythonscriptinterpreter.h>

```

### Public Static Functions

```

std::function<R (Args...)> toNativeFunction
PyObject *self, PyObject *fct
template<class ...Args>
class PyObjectFunctionToNativeFunction<void, Args...>
#include <pythonscriptinterpreter.h>

```

### Public Static Functions

```

std::function<void (Args...)> toNativeFunction
PyObject *self, PyObject *fct
template<class V>
class PyObjectToNative
#include <pythonscriptinterpreter.h> Helps to return a c++ value from a Python object (borrows
the reference)
template<>
class PyObjectToNative<bool>
#include <pythonscriptinterpreter.h>

```

### Public Static Functions

```
bool toNative (PyObject*, PyObject *o)

template<>
class PyObjectToNative<double>
    #include <pythonscriptinterpreter.h>
```

### Public Static Functions

```
double toNative (PyObject*, PyObject *o)

template<>
class PyObjectToNative<int>
    #include <pythonscriptinterpreter.h>
```

### Public Static Functions

```
int toNative (PyObject*, PyObject *o)

template<>
class PyObjectToNative<QByteArray>
    #include <pythonscriptinterpreter.h>
```

### Public Static Functions

```
QByteArray toNative (PyObject*, PyObject *o)

template<>
class PyObjectToNative<qint64>
    #include <pythonscriptinterpreter.h>
```

### Public Static Functions

```
qint64 toNative (PyObject*, PyObject *o)

template<class V>
class PyObjectToNative<QList<V>>
    #include <pythonscriptinterpreter.h>
```

### Public Static Functions

```
QList<V> toNative (PyObject *a, PyObject *o)

template<>
class PyObjectToNative<QString>
    #include <pythonscriptinterpreter.h>
```

### Public Static Functions

```
QString toNative (PyObject*, PyObject *o)

template<>
class PyObjectToNative<quint64>
    #include <pythonscriptinterpreter.h>
```

### Public Static Functions

```
quint64 toNative (PyObject*, PyObject *o)

template<>
class PyObjectToNative<QVariant>
    #include <pythonscriptinterpreter.h>
```

### Public Static Functions

```
QVariant toNative (PyObject*, PyObject *o)

template<class V, class ... Args> function< V(Args...) > >
    #include <pythonscriptinterpreter.h>
```

### Public Static Functions

```
std::function<V (Args...) > toNative
    PyObject *self, PyObject *io

template<class V> shared_ptr< V > >
    #include <pythonscriptinterpreter.h>
```

### Public Static Functions

```
std::shared_ptr<V> toNative (PyObject*, PyObject *o)

template<class V>
class PyObjectToNative<StringMap<V>>
    #include <pythonscriptinterpreter.h>
```

### Public Static Functions

```
StringMap<V> toNative (PyObject *a, PyObject *o)

template<class V>
class PyObjectToNative<V*>
    #include <pythonscriptinterpreter.h>
```

### Public Static Functions

```
V *toNative (PyObject*, PyObject *o)

template<int pos, typename ...Args>
struct PyObjectTupleToNativeTuple
    #include <pythonscriptinterpreter.h>

template<int pos>
struct PyObjectTupleToNativeTuple<pos>
    #include <pythonscriptinterpreter.h>
```

### Public Static Functions

```
std::tuple toNativeTuple (PyObject*, PyObject*)

template<int pos, class T, class ...Args>
struct PyObjectTupleToNativeTuple<pos, T, Args...>
    #include <pythonscriptinterpreter.h>
```

### Public Static Functions

```
std::tuple<T, Args...> toNativeTuple (PyObject *a, PyObject *b)

template<typename D, typename T, T>
struct ScriptedMethodProxy
    #include <pythonscriptinterpreter.h>

template<typename T, typename R, typename ... Args, std::function< R(Args...)> f>
    #include <pythonscriptinterpreter.h>
```

### Public Static Functions

```
void setImplementation (T *inst, std::function<R> Args...
    > vfcn)

PyObject *PyObjectFunctionSetImplementation (PyObject *a, PyObject *b)

namespace search
    Infrastructure for searches in the filesystem.
    See sh::search::SearchManager for more.

class Search : public QObject, public std::enable_shared_from_this<Search>
    #include <search.h> A search object represents one search request the user made.
```

## Public Functions

**~Search** ()

std::shared\_ptr<const *sh::filesystem::Eurl*> **eurl** ()  
Returns the eurl containing the search configuration.

std::shared\_ptr<*sh::filesystem::FilesystemNode*> **searchnode** ()  
Returns the root node of the search (the one named like ‘*Search ...*’).

**Search** (std::shared\_ptr<const *filesystem::Eurl*> *eurl*, std::shared\_ptr<*sh::filesystem::FilesystemNode*>  
*searchnode*)  
See *sh::search::SearchManager::requestSearch*.

void **search** ()  
Start the searching.

bool **isSearching** ()  
Returns if the search is currently in progress.

## Signals

void **isSearchingChanged** ()  
Triggers when *Search::isSearching* changes.

## Private Functions

void **search** (std::shared\_ptr<const *filesystem::Eurl*> *eurl*, QString *relpath*)

void **insertFileItem** (std::shared\_ptr<const *sh::filesystem::Eurl*> *item*, QString *relpath*,  
*sh::filesystem::FilesystemNodeType* *ftype*)

std::shared\_ptr<*sh::filesystem::FilesystemNode*> **getDirItem** (std::shared\_ptr<const  
*sh::filesystem::Eurl*> *item*)

## Private Members

std::shared\_ptr<const *sh::filesystem::Eurl*> **\_eurl**

std::shared\_ptr<*sh::filesystem::FilesystemNode*> **\_searchnode**

*sh::search::SearchConfiguration* \***\_config**

QHash<std::shared\_ptr<const *sh::filesystem::Eurl*>, std::shared\_ptr<*sh::filesystem::FilesystemNode*>> **\_dirs**

QMutex **\_isSearchingMutex**

bool **\_isSearching** = false

bool **\_isSearchAgain** = false

**class SearchConfiguration**

*#include <searchconfiguration.h>* A search configuration.

*Search* configurations contain a list of search criteria and some auxiliary fields. Each search configuration describes what and how a user wants to search (just not where).

## Public Functions

**SearchConfiguration** ()

QString **serialize** ()

Serialize this search configuration to a string. See also *SearchConfiguration::deserialize*.

## Public Members

QList<std::shared\_ptr<sh::search::SearchCriterion>> **criteria**

bool **showAsTree** = false

## Public Static Functions

*SearchConfiguration* \***deserialize** (QString s)

Deserialize a search configuration from a string. See also *SearchConfiguration::serialize*.

**class SearchCriterion**

*#include <searchcriterion.h>* Abstract base class for a search criterion.

Each search criterion implements how a user can filter his files in a search.

Register an implementation, e.g. with ‘REGISTER\_CRITERION\_FACTORY’ of “search/searchcriterionfactoryfromfunction.h”.

Subclassed by *sh::search::criteria::AbstractActionDrivenSearchCriterion*, *sh::search::criteria::AbstractStringSearchCriterion*, *sh::search::criteria::ExtendedAttributeSearchCriterion*, *sh::search::criteria::MtimeSearchCriterion*

## Public Functions

**SearchCriterion** (std::shared\_ptr<sh::search::SearchCriterionFactory> factory)

**~SearchCriterion** ()

QString **serialize** ()

Serializes this criterion to a string. See also *SearchCriterion::deserialize*.

bool **match** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)  
= 0

Checks if a given file (by eurl) matches this criterion. .

QString **valuedescription** () = 0

Returns a human readable text describing the current criterion shortly and precisely. .

std::shared\_ptr<sh::search::SearchCriterionFactory> **factory** ()

Returns the factory this criterion has created (provides some more metadata).

## Public Static Functions

`std::shared_ptr<sh::search::SearchCriterion> deserialize (QString s)`  
 Deserializes this criterion from a string. See also `SearchCriterion::serialize`.

**class SearchCriterionFactory : public** `std::enable_shared_from_this<SearchCriterionFactory>`  
*#include <searchcriterionfactory.h>* Abstract base class for a search criterion factory, which constructs some `sh::search::SearchCriterion` instances.

It also provides some control methods and metadata (which is possible since there is a separate factory for each criterion class).

Note: Though possible in some exotic situations, one should not override this class. Override and register `sh::search::SearchCriterion` instead!

Subclassed by `sh::scripting::api::ApiSearchCriterionFactory`, `sh::search::SearchCriterionFactoryFromFunction`

## Public Functions

**SearchCriterionFactory ()**

`QString key () = 0`  
 Returns the string key for the associated search criterion class. .

`QString description () = 0`  
 Returns the description string for the associated search criterion class. .

`void editor (std::shared_ptr<sh::search::SearchCriterion> c, sh::ui::SearchPanelConfiguration *panelcfg) = 0`  
 Builds the search config ui in a search panel.  
 Takes the data from `c` and populates `panelcfg` with it.

`void editorupdateconfig (sh::ui::SearchPanelConfiguration *panelcfg, std::shared_ptr<sh::search::SearchCriterion> c) = 0`  
 Updates own data with the content from the search config ui in a search panel.  
 Takes the data from `panelcfg` and updates `c` with it.

`std::shared_ptr<sh::search::SearchCriterion> construct () = 0`  
 Constructs an instance of the associated search criterion class. .

`bool isVisibleFor (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::FilesystemNode> node)`  
 Checks if the associated search criterion class should be offered to the user for a given node. .

**~SearchCriterionFactory ()**

**class SearchCriterionFactoryFromFunction : public** `sh::search::SearchCriterionFactory`  
*#include <searchcriterionfactoryfromfunction.h>* A search criterion factory implementation which uses some additional static methods in the `sh::search::SearchCriterion` subclasses for control.

This is the default search criterion factory. It is typically used indirectly by the ‘REGISTER\_CRITERION\_FACTORY’ macro from here.

## Public Functions

**SearchCriterionFactoryFromFunction** (QString *key*, std::function<QString>  
 > *descriptionfct* std::function<void std::shared\_ptr<sh::search::SearchCriterion>,  
 sh::ui::SearchPanelConfiguration\*> *editorfct*, std::function<void sh::ui::SearchPanelConfiguration\*,  
 std::shared\_ptr<sh::search::SearchCriterion>> *editorupdateconfigfct*,  
 std::function<std::shared\_ptr<sh::search::SearchCriterion> std::shared\_ptr<sh::search::SearchCriterionFactory>>>  
*constructfct*)

QString **key** ()  
 Returns the string key for the associated search criterion class. .

QString **description** ()  
 Returns the description string for the associated search criterion class. .

void **editor** (std::shared\_ptr<sh::search::SearchCriterion> *c*,  
 sh::ui::SearchPanelConfiguration \**panelcfg*)  
 Builds the search config ui in a search panel.  
 Takes the data from *c* and populates *panelcfg* with it.

void **editorupdateconfig** (sh::ui::SearchPanelConfiguration \**panelcfg*,  
 std::shared\_ptr<sh::search::SearchCriterion> *c*)  
 Updates own data with the content from the search config ui in a search panel.  
 Takes the data from *panelcfg* and updates *c* with it.

std::shared\_ptr<SearchCriterion> **construct** ()  
 Constructs an instance of the associated search criterion class. .

bool **isVisibleFor** (sh::filesystem::Operation \**op*, std::shared\_ptr<sh::filesystem::FileSystemNode>  
*node*)  
 Checks if the associated search criterion class should be offered to the user for a given node. .

## Private Members

QString **\_key**

std::function<QString ()> **\_descriptionfct**

std::function<void (std::shared\_ptr<sh::search::SearchCriterion>,  
 sh::ui::SearchPanelConfiguration\*)> **\_editorfct**

std::function<void (sh::ui::SearchPanelConfiguration\*, std::shared\_ptr<sh::search::SearchCriterion>)>  
**\_editorupdateconfigfct**

std::function< std::shared\_ptr< sh::search::SearchCriterion > std::shared\_ptr< sh::search::SearchCriterionFactory >> **\_constructfct**

**class SearchFilesystemHandler**: public sh::filesystem::FilesystemHandler, public sh::base::Singleton  
*#include <searchfilesystemhandler.h>* A filesystem handler for presenting filesystem search results.



## Public Functions

```

void itemList (sh::filesystem::Operation *op, std::shared_ptr<const
             sh::filesystem::Eurl> eurl, sh::filesystem::FilesystemNodeType type,
             sh::filesystem::FilesystemNodeListEditor *list) override

sh::filesystem::FilesystemNodeType getType (sh::filesystem::Operation *op,
             std::shared_ptr<const sh::filesystem::Eurl>
             eurl) override

qint64 getSize (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
             eurl) override

QDateTime getMtime (sh::filesystem::Operation *op, std::shared_ptr<const
             sh::filesystem::Eurl> eurl) override

void setMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
             eurl, QDateTime time) override

QString getMimeType (sh::filesystem::Operation *op, std::shared_ptr<const
             sh::filesystem::Eurl> eurl) override

QString getLinkTarget (sh::filesystem::Operation *op, std::shared_ptr<const
             sh::filesystem::Eurl> eurl) override

bool canGetFileContent (sh::filesystem::Operation *op, std::shared_ptr<const
             sh::filesystem::Eurl> eurl) override

std::shared_ptr<QIODevice> getFileContent (sh::filesystem::Operation
             *op, std::shared_ptr<const
             sh::filesystem::Eurl> eurl) override

bool canCreateDirectory (sh::filesystem::Operation *op, std::shared_ptr<const
             sh::filesystem::Eurl> eurl) override

void createDirectory (sh::filesystem::Operation *op, std::shared_ptr<const
             sh::filesystem::Eurl> eurl) override

bool canCreateLink (sh::filesystem::Operation *op, std::shared_ptr<const
             sh::filesystem::Eurl> eurl) override

void createLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
             eurl, QString target) override

bool canCreateFile (sh::filesystem::Operation *op, std::shared_ptr<const
             sh::filesystem::Eurl> eurl) override

void createFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
             eurl, QIODevice *content, HandlerTransfer *handlertransfer) override

bool canDeleteItem (sh::filesystem::Operation *op, std::shared_ptr<const
             sh::filesystem::Eurl> eurl) override

void deleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
             eurl) override

bool canRenameItem (sh::filesystem::Operation *op, std::shared_ptr<const
             sh::filesystem::Eurl> src) override

void renameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
             src, QString destpath) override

QList<std::shared_ptr<sh::actions::AbstractActionItem>> getActions (const
             QList<std::shared_ptr<const
             sh::filesystem::Eurl>>
             eurls) override

```

```
void customizeUi (std::shared_ptr<sh::filesystem::FilesystemNode> node, bool *showSearch-
                  Panel) override

void configureItems (sh::filesystem::Operation *op, QList<std::shared_ptr<sh::filesystem::FilesystemNode>>
                     nodes) override
    Configure newly created nodes (e.g. setting another icon or changing the display name).

void itemlist (sh::filesystem::Operation *op, std::shared_ptr<const
              sh::filesystem::Eurl> eurl, sh::filesystem::FilesystemNodeType type,
              sh::filesystem::FilesystemNodeListEditor *list) = 0
    Determine a list of subelements in a certain directory.

sh::filesystem::FilesystemNodeType getType (sh::filesystem::Operation *op,
                                             std::shared_ptr<const sh::filesystem::Eurl>
                                             eurl) = 0
    Determine if a file is regular, or a dir, or a link, ...

qint64 getSize (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
               eurl) = 0
    Determine the size of a file.

QDateTime getMtime (sh::filesystem::Operation *op, std::shared_ptr<const
                   sh::filesystem::Eurl> eurl) = 0
    Determine the mtime of a file.

void setMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
               eurl, QDateTime time) = 0
    Set the mtime of a file.

QString getMimeType (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) = 0
    Determine mime type of a file.

QString getLinkTarget (sh::filesystem::Operation *op, std::shared_ptr<const
                      sh::filesystem::Eurl> eurl) = 0
    Resolve a link.

QList<QString> listExtendedAttributes (sh::filesystem::Operation *op,
                                       std::shared_ptr<const sh::filesystem::Eurl>
                                       eurl)
    Fetches the list of available extended attributes for an entry.

quint64 getExtendedAttributeSize (sh::filesystem::Operation *op,
                                   std::shared_ptr<const sh::filesystem::Eurl>
                                   eurl, QString attribute)
    Returns the size of value for one extended attribute for an entry.

QByteArray getExtendedAttribute (sh::filesystem::Operation *op,
                                  std::shared_ptr<const sh::filesystem::Eurl>
                                  eurl, QString attribute)
    Returns the value for one extended attribute for an entry.

void setExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
                           sh::filesystem::Eurl> eurl, QString attribute, QByteArray
                           value)
    Sets the value for one extended attribute for an entry.

void removeExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
                              sh::filesystem::Eurl> eurl, QString attribute)
    Removes one extended attributes for an entry.

QMap<QString, QString> getCustomAttributes (sh::filesystem::Operation
                                              *op, std::shared_ptr<const
                                              sh::filesystem::Eurl> eurl)
```

Returns the custom attributes for an entry.

In contrast to extended attributes, the custom attributes are not directly stored in the filesystem this way, but handle implementation specific aspects (like permissions).

```
void setCustomAttribute (sh::filesystem::Operation *op,      std::shared_ptr<const
                        sh::filesystem::Eurl> eurl, QString attribute, QString value)
```

Sets the value for one custom attribute for an entry.

See also `getCustomAttributes`.

```
bool canGetFileContent (sh::filesystem::Operation *op,      std::shared_ptr<const
                        sh::filesystem::Eurl> eurl) = 0
```

Returns if it is allowed to get the content of a certain file.

```
std::shared_ptr<QIODevice> getFileContent (sh::filesystem::Operation
                                           *op,      std::shared_ptr<const
                                           sh::filesystem::Eurl> eurl) = 0
```

Get the content of a file.

```
bool canCreateDirectory (sh::filesystem::Operation *op,      std::shared_ptr<const
                        sh::filesystem::Eurl> eurl) = 0
```

Returns if it is allowed to create subdirectories in a certain directory.

```
void createDirectory (sh::filesystem::Operation *op,      std::shared_ptr<const
                        sh::filesystem::Eurl> eurl) = 0
```

Create a directory.

```
bool canCreateLink (sh::filesystem::Operation *op,      std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) = 0
```

Returns if it is allowed to create a link in a certain directory.

```
void createLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                 eurl, QString target) = 0
```

Create a link.

```
bool canCreateFile (sh::filesystem::Operation *op,      std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) = 0
```

Returns if it is allowed to create files in a certain directory.

```
void createFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                 eurl, QIODevice *content, HandlerTransfer *handlertransfer) = 0
```

Creates a file with some content.

It may use `handlertransfer` for some better integration, like cancel support and progress monitoring.

```
bool canDeleteItem (sh::filesystem::Operation *op,      std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) = 0
```

Returns if it is allowed to delete a certain file/directory/link/...

```
void deleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                 eurl) = 0
```

Delete a file/directory/link/...

```
void deleteDirectoryIfEmpty (sh::filesystem::Operation *op, std::shared_ptr<const
                             sh::filesystem::Eurl> eurl)
```

Deletes a directory only if it is empty.

```
bool canRenameItem (sh::filesystem::Operation *op,      std::shared_ptr<const
                    sh::filesystem::Eurl> src) = 0
```

Returns if it is allowed to rename a certain file, directory, link, ... (see `renameItem`).

void **renameItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*>  
src, QString destpath) = 0  
Renames an item to another path.

It can be a simple filename change or also changes in the deeper path segments.

QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> **getActions** (const  
QList<std::shared\_ptr<const  
*sh::filesystem::Eurl*>>  
*eurls*) = 0

Returns a list of actions (see former example) for showing for certain files.

bool **requiresResolvedLinks** ()  
Returns if this handler requires to be used by the engine with resolved links.

void **viewEnteredDirectory** (std::shared\_ptr<const *sh::filesystem::Eurl*>)  
Callback for preparing stuff when the view entered some directory (e.g. for enabling watchers).  
See also viewLeftDirectory.

void **viewLeftDirectory** (std::shared\_ptr<const *sh::filesystem::Eurl*>)  
Callback for preparing stuff when the view left some directory. See also viewEnteredDirectory.

bool **isWellKnownScheme** ()  
Returns if the scheme for this handler is a well known one (which external programs typically  
would understand).

void **search** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl*> eurl, std::function<void> std::shared\_ptr<const  
*sh::filesystem::Eurl*>, QList<std::shared\_ptr<*sh::search::SearchCriterion*>>,  
*sh::filesystem::FilesystemNodeType* ftype  
> addfile, std::function<void>std::shared\_ptr<const *sh::filesystem::Eurl*>> visitdir,  
QList<std::shared\_ptr<*sh::search::SearchCriterion*>> criteriaHelps searching for files.

Override this method with a behavior identical to the default, if a specialized implementation can  
be much faster than the default one.

void **customizeUi** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> node, bool \*showSearch-  
Panel)  
Creates a custom gui, which becomes part of the fileview.

*sh::filesystem::FilesystemModel* \***model** ()  
A convenience shortcut to the *sh::filesystem::FilesystemModel* (a singleton).

void **doShutdown** ()  
Executes singleton shutdown.

void **shutdown** ()  
Shutdown down this singleton.

bool **isAlive** ()  
Returns if this singleton is alive (true until its shutdown begins).

## Private Functions

**SearchFilesystemHandler** ()

**class SearchManager** : public QObject, public *sh::base::Singleton*  
*#include <searchmanager.h>* Manager for searches in the filesystem.

## Public Functions

std::shared\_ptr<*sh::search::Search*> **requestSearch** (std::shared\_ptr<const  
*sh::filesystem::Eurl*> *eurl*,  
 QString *searchconfig*,  
 std::shared\_ptr<*sh::filesystem::FilesystemNode*>  
*searchrootnode*)

Requests a search.

void **removeSearch** (std::shared\_ptr<*sh::search::Search*> *search*)  
 Removes a search.

std::shared\_ptr<*sh::search::Search*> **findSearch** (std::shared\_ptr<const  
*sh::filesystem::Eurl*> *eurl*)  
 Returns the *Search* instance for an eurl for a search root node.

void **addFactory** (int *index*, std::shared\_ptr<*SearchCriterionFactory*> *f*)  
 Adds a search criterion factory.

QList<std::shared\_ptr<*SearchCriterionFactory*>> **factories** ()  
 Returns the sorted list of search criterion factories.

**~SearchManager** ()

void **doShutdown** ()  
 Executes singleton shutdown.

void **shutdown** ()  
 Shutdown down this singleton.

bool **isAlive** ()  
 Returns if this singleton is alive (true until its shutdown begins).

## Public Static Attributes

const int **REGISTER\_SEARCHCRITERION\_INDEX\_CORE** = 1000000  
 Base value for display indexes of core (i.e. very important) search criteria.  
 Used in *SearchManager::addFactory*.

const int **REGISTER\_SEARCHCRITERION\_INDEX\_NORMAL** = 2000000  
 Base value for display indexes of search criteria with normal importance.  
 Used in *SearchManager::addFactory*.

const int **REGISTER\_SEARCHCRITERION\_INDEX\_EXOTIC** = 3000000  
 Base value for display indexes of exotic search criteria.  
 Used in *SearchManager::addFactory*.

## Private Functions

**SearchManager** ( )

## Private Members

QList<std::shared\_ptr<*sh::search::Search*>> **\_searches**

QList<std::weak\_ptr<*sh::search::Search*>> **\_weak\_searches**

QMap<int, std::shared\_ptr<*SearchCriterionFactory*>> **\_factories**

### namespace **criteria**

Implementations of search criteria.

Subclasses of *sh::search::SearchCriterion*.

**class AbstractActionDrivenSearchCriterion** : public *sh::search::SearchCriterion*  
*#include <abstractactiondrivensearchcriterion.h>* Abstract search criterion filtering by something which is configured in a wizard-like way by an action execution.

This provides an easy yet powerful programming interface, e.g. for usage in scripting.

Subclassed by *sh::scripting::api::ApiSearchCriterion*

## Public Functions

**AbstractActionDrivenSearchCriterion** (std::shared\_ptr<*sh::search::SearchCriterionFactory*>  
*factory*)

bool **match** (*sh::filesystem::Operation* \**op*, std::shared\_ptr<const *sh::filesystem::Eurl*>  
*eurl*)

Checks if a given file (by eurl) matches this criterion. .

bool **match2** (*sh::filesystem::Operation* \**op*, std::shared\_ptr<const *sh::filesystem::Eurl*>  
*eurl*) = 0

QString **valuedescription** ( )

Returns a human readable text describing the current criterion shortly and precisely. .

QString **serialize** ( )

Serializes this criterion to a string. See also *SearchCriterion::deserialize*.

std::shared\_ptr<*sh::search::SearchCriterionFactory*> **factory** ( )

Returns the factory this criterion has created (provides some more metadata).

## Public Members

QStringList **searchspec**

## Public Static Functions

```
void _createEditor (std::shared_ptr<sh::search::SearchCriterion>          cfg,
                   sh::ui::SearchPanelConfiguration *panelcfg)
void _editorUpdateConfiguration (sh::ui::SearchPanelConfiguration *panelcfg,
                                  std::shared_ptr<sh::search::SearchCriterion> c)

std::shared_ptr<sh::search::SearchCriterion> deserialize (QString s)
    Deserializes this criterion from a string. See also SearchCriterion::serialize.
```

## Friends

```
friend class ActionAbstractActionDrivenSearchCriterionConfigure

class AbstractStringSearchCriterion : public sh::search::SearchCriterion
    #include <abstractstringsearchcriterion.h> Abstract search criterion filtering by some string
    value (providing different modes like exact, fuzzy, regexp, ...).

    Subclassed          by          sh::search::criteria::FileContentSearchCriterion,
    sh::search::criteria::FilenameSearchCriterion
```

## Public Types

```
enum Mode
    Values:

    enumerator Simple
    enumerator Exact
    enumerator RegExp
    enumerator Fuzzy
```

## Public Functions

```
AbstractStringSearchCriterion (std::shared_ptr<sh::search::SearchCriterionFactory>
                                factory)

bool match (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
            eurl)
    Checks if a given file (by eurl) matches this criterion. .

QString valuedescription ()
    Returns a human readable text describing the current criterion shortly and precisely. .

QString serialize ()
    Serializes this criterion to a string. See also SearchCriterion::deserialize.

std::shared_ptr<sh::search::SearchCriterionFactory> factory ()
    Returns the factory this criterion has created (provides some more metadata).
```

## Public Members

QString **string**

*Mode* **mode** = *Mode::Simple*

## Public Static Functions

void **\_createEditor** (std::shared\_ptr<*sh::search::SearchCriterion*> *cfg*,  
                    *ui::SearchPanelConfiguration* \**panelcfg*)

void **\_editorUpdateConfiguration** (*sh::ui::SearchPanelConfiguration* \**panelcfg*,  
                                    std::shared\_ptr<*sh::search::SearchCriterion*> *c*)

int **levenshteinSubstringDistance** (QString *needle*, QString *haystack*)

double **relativeLevenshteinPartsSubstringDistance** (QString *needle*,  
  QString *haystack*)

std::shared\_ptr<*sh::search::SearchCriterion*> **deserialize** (QString *s*)  
Deserializes this criterion from a string. See also *SearchCriterion::serialize*.

## Public Static Attributes

const QStringList **ModeCodes**

**class ActionAbstractActionDrivenSearchCriterionConfigure** : public *sh::actions::ActionActionDrivenSearchCriterionConfigure*  
#include <abstractactiondrivensearchcriterion.h>

## Public Functions

**ActionAbstractActionDrivenSearchCriterionConfigure** (std::shared\_ptr<*AbstractActionDrivenSearchCriterionConfigure*> *cfg*,  
  QMutex \**mutex*)

void **action** (*sh::actions::ActionExecutionInfo* \**info*)  
The action implementation, i.e. what the actions should actually do.

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* =  
  false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.



bool **enabled** ()  
 Checks if this action is enabled.

bool **isChecked** ()  
 Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
 Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
 Returns the precedence for becoming the default action of a parent submenu.  
 This e.g. leads to a bold label.  
 The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
 Sets the displayed text.

void **setIcon** (QString *icon*)  
 Sets the icon.

void **setEnabled** (bool *enabled*)  
 Sets if the item is enabled.

void **setChecked** (bool *checked*)  
 Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
 Sets the visibility of this item.

bool **visible** ()  
 Checks the visibility of this item (non-recursively).

std::weak\_ptr<AbstractActionItem> **parentAction** ()  
 Returns the parent action, if it is added to a container.

void **\_\_setParentAction** (std::shared\_ptr<AbstractActionItem> *parent*)  
 Sets the parent action. .

void **initializeAsync** (std::function<void>  
 > *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
 Synchronously initializes the action.

bool **isInitialized** ()  
 Checks if this action is initialized.

bool **isInitializing** ()  
 Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

## Private Members

std::shared\_ptr<AbstractActionDrivenSearchCriterion> **\_cfg**

QMutex **\*\_mutex**

**class ExtendedAttributeSearchCriterion** : public [sh::search::SearchCriterion](#)  
*#include <extendedattributearchcriterion.h>* [Search](#) criterion filtering by extended attributes.

## Public Types

enum **Mode**

*Values:*

enumerator **None**

enumerator **Simple**

enumerator **RegExp**

enumerator **Fuzzy**

## Public Functions

**ExtendedAttributeSearchCriterion** (std::shared\_ptr<[sh::search::SearchCriterionFactory](#)>  
*factory*)

bool **match** ([sh::filesystem::Operation](#) \**op*, std::shared\_ptr<const [sh::filesystem::Eurl](#)>  
*eurl*)  
Checks if a given file (by eurl) matches this criterion. .

QString **valuedescription** ()  
Returns a human readable text describing the current criterion shortly and precisely. .

QString **serialize** ()  
Serializes this criterion to a string. See also [SearchCriterion::deserialize](#).

std::shared\_ptr<[sh::search::SearchCriterionFactory](#)> **factory** ()  
Returns the factory this criterion has created (provides some more metadata).

## Public Members

*Mode* **keymode** = *Mode::None*

*Mode* **valuemode** = *Mode::Simple*

QString **key**

QString **value**

## Public Static Functions

```
void createEditor (std::shared_ptr<sh::search::SearchCriterion>                cfg,
                  sh::ui::SearchPanelConfiguration *panelcfg)
void editorUpdateConfiguration (sh::ui::SearchPanelConfiguration *panelcfg,
                                std::shared_ptr<sh::search::SearchCriterion> c)

void doInitialize ()
void doShutdown ()

std::shared_ptr<sh::search::SearchCriterion> deserialize (QString s)
    Deserializes this criterion from a string. See also SearchCriterion::serialize.
```

## Public Static Attributes

```
const QStringList ModeCodes
```

```
class FileContentSearchCriterion: public sh::search::criteria::AbstractStringSearchCriterion
    #include <filecontentsearchcriterion.h> Search criterion filtering by file contents.
```

## Public Types

```
enum Mode
    Values:

    enumerator Simple
    enumerator Exact
    enumerator RegExp
    enumerator Fuzzy
```

## Public Functions

```
FileContentSearchCriterion (std::shared_ptr<sh::search::SearchCriterionFactory>
                             factory)

bool match (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
            eurl)
    Checks if a given file (by eurl) matches this criterion. .

QString valuedescription ()
    Returns a human readable text describing the current criterion shortly and precisely. .

QString serialize ()
    Serializes this criterion to a string. See also SearchCriterion::deserialize.

std::shared_ptr<sh::search::SearchCriterionFactory> factory ()
    Returns the factory this criterion has created (provides some more metadata).
```

## Public Members

QString **string**

*Mode* **mode** = *Mode::Simple*

## Public Static Functions

void **createEditor** (std::shared\_ptr<*sh::search::SearchCriterion*> *cfg*,  
                  *sh::ui::SearchPanelConfiguration* \*panelcfg)

void **editorUpdateConfiguration** (*sh::ui::SearchPanelConfiguration* \*panelcfg,  
                                  std::shared\_ptr<*sh::search::SearchCriterion*> c)

void **doInitialize** ()

void **doShutdown** ()

void **\_createEditor** (std::shared\_ptr<*sh::search::SearchCriterion*> *cfg*,  
                    *ui::SearchPanelConfiguration* \*panelcfg)

void **\_editorUpdateConfiguration** (*sh::ui::SearchPanelConfiguration* \*panelcfg,  
                                  std::shared\_ptr<*sh::search::SearchCriterion*> c)

int **levenshteinSubstringDistance** (QString *needle*, QString *haystack*)

double **relativeLevenshteinPartsSubstringDistance** (QString *needle*,  
  QString *haystack*)

std::shared\_ptr<*sh::search::SearchCriterion*> **deserialize** (QString *s*)  
    Deserializes this criterion from a string. See also *SearchCriterion::serialize*.

## Public Static Attributes

const QStringList **ModeCodes**

**class FilenameSearchCriterion**: public *sh::search::criteria::AbstractStringSearchCriterion*  
    #include <filenamesearchcriterion.h> *Search* criterion filtering by file names.

## Public Types

enum **Mode**

*Values:*

enumerator **Simple**

enumerator **Exact**

enumerator **RegExp**

enumerator **Fuzzy**

## Public Functions

**FilenameSearchCriterion** (std::shared\_ptr<sh::search::SearchCriterionFactory> *factory*)

bool **match** (sh::filesystem::Operation \**op*, std::shared\_ptr<const sh::filesystem::Eurl> *eurl*)

Checks if a given file (by eurl) matches this criterion. .

QString **valuedescription** ()

Returns a human readable text describing the current criterion shortly and precisely. .

QString **serialize** ()

Serializes this criterion to a string. See also *SearchCriterion::deserialize*.

std::shared\_ptr<sh::search::SearchCriterionFactory> **factory** ()

Returns the factory this criterion has created (provides some more metadata).

## Public Members

QString **string**

*Mode* **mode** = *Mode::Simple*

## Public Static Functions

void **createEditor** (std::shared\_ptr<sh::search::SearchCriterion> *cfg*,  
sh::ui::SearchPanelConfiguration \**panelcfg*)

void **editorUpdateConfiguration** (sh::ui::SearchPanelConfiguration \**panelcfg*,  
std::shared\_ptr<sh::search::SearchCriterion> *c*)

void **doInitialize** ()

void **doShutdown** ()

void **\_createEditor** (std::shared\_ptr<sh::search::SearchCriterion> *cfg*,  
ui::SearchPanelConfiguration \**panelcfg*)

void **\_editorUpdateConfiguration** (sh::ui::SearchPanelConfiguration \**panelcfg*,  
std::shared\_ptr<sh::search::SearchCriterion> *c*)

int **levenshteinSubstringDistance** (QString *needle*, QString *haystack*)

double **relativeLevenshteinPartsSubstringDistance** (QString *needle*,  
QString *haystack*)

std::shared\_ptr<sh::search::SearchCriterion> **deserialize** (QString *s*)

Deserializes this criterion from a string. See also *SearchCriterion::serialize*.

## Public Static Attributes

**const** QStringList **ModeCodes**

**class** **MtimeSearchCriterion**: **public** *sh::search::SearchCriterion*  
*#include <mtimesearchcriterion.h>* *Search* criterion filtering by file modification times.

## Public Functions

**MtimeSearchCriterion** (std::shared\_ptr<*sh::search::SearchCriterionFactory*> *factory*)

**bool** **match** (*sh::filesystem::Operation* \**op*, std::shared\_ptr<**const** *sh::filesystem::Eurl*>  
*eurl*)  
Checks if a given file (by eurl) matches this criterion. .

QString **valuedescription** ()  
Returns a human readable text describing the current criterion shortly and precisely. .

QString **serialize** ()  
Serializes this criterion to a string. See also *SearchCriterion::deserialize*.

std::shared\_ptr<*sh::search::SearchCriterionFactory*> **factory** ()  
Returns the factory this criterion has created (provides some more metadata).

## Public Members

QDateTime **from**

QDateTime **to**

## Public Static Functions

**void** **createEditor** (std::shared\_ptr<*sh::search::SearchCriterion*> *cfg*,  
*sh::ui::SearchPanelConfiguration* \**panelcfg*)

**void** **editorUpdateConfiguration** (*sh::ui::SearchPanelConfiguration* \**panelcfg*,  
std::shared\_ptr<*sh::search::SearchCriterion*> *c*)

**void** **doInitialize** ()

**void** **doShutdown** ()

std::shared\_ptr<*sh::search::SearchCriterion*> **deserialize** (QString *s*)  
Deserializes this criterion from a string. See also *SearchCriterion::serialize*.

## namespace settings

Infrastructure for settings, so everything you see in ‘Manage saved settings’.

See the abstract base class *sh::settings::Setting* and *sh::settings::SettingsManager* for more.

## Enums

### enum SettingGroup

Groups of settings.

A choice here does not make a logical impact. It is merely a matter of graphical presentation.

*Values:*

**enumerator** GLOBAL = 1

**enumerator** GUI = 2

**enumerator** FILEHANDLING = 3

**enumerator** DATAVIEW = 4

**enumerator** BEHAVIOR = 5

**enumerator** SPECIAL = 6

### class ProfileNode

*#include <profilenode.h>* One entry in a Shallot settings profile, so one item as selectable in the 'Manage settings' dialog.

## Public Functions

### ProfileNode ()

Constructed only by the infrastructure and made available otherwise.

### QString nodeId ()

An identifier name.

### void setNodeId (QString nodeId)

Sets the identifier name.

### std::shared\_ptr<const filesystem::Eurl> eurl ()

The eurl of the directory, or 0 for global profile nodes.

### void setEurl (std::shared\_ptr<const filesystem::Eurl> eurl)

Sets the eurl.

### QString profilename ()

The profile name.

### void setProfilename (QString profilename)

Sets the profile name.

### QStringList inheritsFrom ()

List of profile names from which this node inherits.

### void setInheritsFrom (QStringList profilenames)

Sets list of profile names from which this node inherits.

### bool withSubfolders ()

If this node also applies recursively on subfolders.

### void setWithSubfolders (bool v)

Sets of this node also applies recursively on subfolders.

### void setSetting (QString name, QString value, int onlyInFileview = -1)

Stores a value for a setting for applying it later.

QList<QString> **getSettingNames** ()

List of settings names which are set in this node.

QString **getSettingValue** (QString *name*)

Gets the stored value of a setting by setting name.

int **getSettingOnlyInFileviewIndex** (QString *name*)

Gets if a setting applies to a certain fileview or to the entire window.

### Private Members

QMap<QString, QString> **\_values**

QMap<QString, int> **\_valuesOnlyInFileviewIndex**

std::shared\_ptr<const *sh::filesystem::Eurl*> **\_eurl**

QString **\_profilename**

QStringList **\_inheritsFrom**

bool **\_withSubfolders**

QString **\_nodeId**

### class Setting

*#include <setting.h>* Abstract base class for a Shallot setting.

Those have the greatest flexibility. They have to be stored manually by the user with many options. See Shallot documentation for details.

Use *sh::settings::SettingsRegistration* for registering an implementation.

Subclassed by *sh::scripting::api::ApiSetting*, *sh::settings::common::FileDetailsPanelVisible*,  
*sh::settings::common::FileViewIconListThumbDimension*, *sh::settings::common::FileViewPath*,  
*sh::settings::common::FileViewViewmode*, *sh::settings::common::JumpbarMode*,  
*sh::settings::common::NumberOfFilePanels*, *sh::settings::common::ShowHiddenFiles*,  
*sh::settings::common::ShowTree*, *sh::settings::common::SizeFormattingMode*,  
*sh::settings::common::StickyTreeview*, *sh::settings::common::WindowTitle*

### Public Functions

**Setting** ()

Is (for subclasses) intended to be directly constructed from everywhere or by registering a factory somewhere.

**~Setting** ()

QString **name** () = 0

Gets the internal name.

QString **description** () = 0

Gets the description text.

*SettingGroup* **group** () = 0

Gets the group;.

bool **isAdvancedSetting** () = 0

Is this an advanced setting?

void **setValue** (*sh::ui::FileView\**, QString)

Called from Shallot core when the value was set (for a per-fileview setting).



void **setValue** (QString)

Called from Shallot core when the value was set (for a not-per-fileview setting).

bool **isGlobal** () = 0

Does this setting apply globally or just for a certain subtree of nodes?

bool **isPerFileview** () = 0

Does this setting apply for each fileview individually or for the complete main window?

QString **getValue** (*sh::ui::FileView* \*filelist) = 0

Get the currently set value.

QString **valueDescription** (QString value)

Gets a human readable description text for a value.

## Public Static Functions

QString **getGroupDescription** (int g)

Low-level function which gets the description text of a group.

**class SettingsManager** : public QObject, public *sh::base::Singleton*

*#include <settingsmanager.h>* Manager for Shallot settings.

This is the more flexible way of shallot customization. It includes everything you see in the ‘Manage saved settings’ or ‘Save settings’.

See *sh::settings::Setting* for more.

## Public Functions

void **applyPerEurlSettingsToFileView** (*sh::ui::FileView* \*list)

Applies all stored settings, which are per-eurl to a fileview.

void **applyGlobalSettingsToFileView** (*sh::ui::FileView* \*list)

Applies all stored settings, which are global to a fileview.

void **applyPerEurlSettingsToMainWindow** ()

Applies all stored settings, which are per-eurl to main window.

void **applyGlobalSettingsToMainWindow** ()

Applies all stored settings, which are global to a main window.

QList<*sh::settings::ProfileNode*\*> **getNodesForProfile** (QString profile)

Returns a list of all stored settings (as *ProfileNodes*) for a given profile.

*sh::settings::ProfileNode* \***getNodeById** (QString nodeId)

Searches and returns a *ProfileNode* by id.

std::shared\_ptr<*sh::settings::Setting*> **getSettingByName** (QString name)

Searches and returns a *Setting* by name.

QList<std::shared\_ptr<*sh::settings::Setting*>> **getAllSettings** ()

Returns a list of all available settings, primarily sorted by group.

QStringList **getProfileList** ()

Returns a list of all existing profiles.

void **removeProfile** (QString profile)

Removes a profile (including all *ProfileNodes* inside it).

void **removeNode** (QString *nodeId*)  
Removes a *ProfileNode* by id.

void **storeProfile** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*, QString *profilename*,  
QMap<QString, QString> *values*, QMap<QString, int> *onlyinfileviewindex*, bool *withSubfolders*, QStringList *inheritFrom*)  
Stores one setting into a profile.

**~SettingsManager** ()

void **doInitialize** ()  
Executes singleton initialization.

void **doShutdown** ()  
Executes singleton shutdown.

void **shutdown** ()  
Shutdown down this singleton.

bool **isAlive** ()  
Returns if this singleton is alive (true until its shutdown begins).

## Signals

void **profilesChanged** ()  
Triggered when the list of profiles or the nodes inside it change.  
  
Can also be triggered when nothing really changed!

## Private Functions

void **applyToFileView** (*sh::ui::FileView* \**list*, QMap<QString, QString> *settings*)  
Applies some settings (as string tuples) to a fileview.

void **applyToMainWindow** (QMap<QString, QString> *settings*)  
Applies some settings (as string tuples) to main window.

QMap<QString, QString> **\_getSettings** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*,  
QString *profilename*, int *fileviewindex*, bool *directNode* = true)  
Computes all effective settings (including inheritance and sub-directories) for a given situation (global or *eurl*-bound; fileview- or mainwindow-bound) and returns it as string tuples.

### Parameters

- *eurl*: The directory to consider as current (or `nullptr` for global settings).
- *profilename*: The current profile.
- *fileviewindex*: The fileview index (or -1 for mainwindow-bound settings).
- *directNode*: If the given directory is to be considered as the real current directory (not e.g. a parent).

QList<*\_SettingsFinderStructure*> **\_getSettings\_flat** (std::shared\_ptr<const  
*sh::filesystem::Eurl*> *eurl*, QString  
*profilename*)  
  
Used internally by *\_getSettings*().

void **invalidateCache** ()  
Invalidates the *\_getSettings* cache.

void **readStoredProfiles** ()  
Reads all profiles (including all setting nodes) from filesystem.

**SettingsManager** ( )

### Private Members

QMap<QString, QMap<QString, QString>> **\_getSettings\_cache**

QMap<QString, std::shared\_ptr<*sh::settings::Setting*>> **\_settings**

QMap<std::shared\_ptr<const *sh::filesystem::Eurl*>, QMap<QString, QList<*sh::settings::ProfileNode*\*>>> **\_profiles**

QMap<QString, *sh::settings::ProfileNode*\*> **\_profileNodesById**

QList<QString> **\_profileNames**

### Friends

**friend class** *sh::settings::SettingsRegistration*

**class** **\_MyHandler** : **public** QXmlDefaultHandler

Used internally for reading profile node XMLs.

### Public Functions

**\_MyHandler** (*sh::settings::ProfileNode* \**node*, QString *filename*)

bool **startElement** (const QString &*namespaceURI*, const QString &*localName*,  
const QString &*qName*, const QXmlAttributes &*atts*)

bool **endElement** (const QString &*namespaceURI*, const QString &*localName*, const  
QString &*qName*)

### Private Functions

void **throwError** ( )

### Private Members

bool **\_stateStarted** = true

bool **\_stateInProfile** = false

*sh::settings::ProfileNode* \* **\_node**

QString **\_filename**

**class** **\_SettingsFinderStructure**

Used internally by *\_getSettings()*.

### Public Functions

```
_SettingsFinderStructure (QMap<QString, QString> settings, QMap<QString, int>  
                           onlyinfileviewindexes, QStringList inheritFrom, bool  
                           withSubfolders)  
  
_SettingsFinderStructure ()
```

### Public Members

```
QMap<QString, QString> _settings  
QMap<QString, int> _onlyinfileviewindexes  
QStringList _inheritFrom  
bool _withSubfolders  
int _onlyFileviewIndex
```

### class SettingsRegistration

*#include <settingsregistration.h>* Used for registering a setting instance.

Constructing a *SettingsRegistration* automatically registers a *sh::settings::Setting*, deleting a one automatically unregisters it.

### Public Functions

```
SettingsRegistration (std::shared_ptr<sh::settings::Setting> setting)  
    Is intended to be directly constructed from everywhere.  
  
~SettingsRegistration ()  
  
std::shared_ptr<sh::settings::Setting> setting ()
```

### Private Members

```
std::shared_ptr<sh::settings::Setting> _setting
```

### namespace common

Implementations of settings.

Subclasses of *sh::settings::Setting* (and possibly some auxiliary stuff). Everything here could be listed in the ‘Save settings’ dialog.

```
class FileDetailsPanelVisible : public sh::settings::Setting  
    #include <filedetailspanelvisible.h>
```

## Public Functions

**FileDetailsPanelVisible** ()

QString **name** ()

Gets the internal name.

QString **description** ()

Gets the description text.

*sh::settings::SettingGroup* **group** ()

Gets the group;.

bool **isAdvancedSetting** ()

Is this an advanced setting?

void **setValue** (QString *value*)

Called from Shallot core when the value was set (for a not-per-fileview setting).

QString **getValue** (*sh::ui::FileView \*filelist*)

Get the currently set value.

bool **isGlobal** ()

Does this setting apply globally or just for a certain subtree of nodes?

bool **isPerFileview** ()

Does this setting apply for each fileview individually or for the complete main window?

QString **valueDescription** (QString *value*)

Gets a human readable description text for a value.

void **setValue** (*sh::ui::FileView\**, QString)

Called from Shallot core when the value was set (for a per-fileview setting).

## Public Static Functions

QString **getGroupDescription** (int *g*)

Low-level function which gets the description text of a group.

```
class FileViewIconListThumbDimension : public sh::settings::Setting
#include <fileviewiconlistthumbdimension.h>
```

## Public Functions

**FileViewIconListThumbDimension** ()

QString **name** ()

Gets the internal name.

QString **description** ()

Gets the description text.

*sh::settings::SettingGroup* **group** ()

Gets the group;.

bool **isAdvancedSetting** ()

Is this an advanced setting?

void **setValue** (*sh::ui::FileView \*filelist*, QString *value*)

Called from Shallot core when the value was set (for a per-fileview setting).

QString **getValue** (*sh::ui::FileView \*filelist*)

Get the currently set value.

bool **isGlobal** ()

Does this setting apply globally or just for a certain subtree of nodes?

bool **isPerFileview** ()

Does this setting apply for each fileview individually or for the complete main window?

QString **valueDescription** (QString *value*)

Gets a human readable description text for a value.

void **setValue** (QString)

Called from Shallot core when the value was set (for a not-per-fileview setting).

## Public Static Functions

QString **getGroupDescription** (int *g*)

Low-level function which gets the description text of a group.

```
class FileViewPath : public sh::settings::Setting
#include <fileviewpath.h>
```

## Public Functions

**FileViewPath** ()

QString **name** ()

Gets the internal name.

QString **description** ()

Gets the description text.

*sh::settings::SettingGroup* **group** ()

Gets the group;.

bool **isAdvancedSetting** ()

Is this an advanced setting?

void **setValue** (*sh::ui::FileView \*fileview*, QString *value*)

Called from Shallot core when the value was set (for a per-fileview setting).

QString **getValue** (*sh::ui::FileView \*filelist*)

Get the currently set value.

bool **isGlobal** ()

Does this setting apply globally or just for a certain subtree of nodes?

bool **isPerFileview** ()

Does this setting apply for each fileview individually or for the complete main window?

void **setValue** (QString)

Called from Shallot core when the value was set (for a not-per-fileview setting).

QString **valueDescription** (QString *value*)

Gets a human readable description text for a value.

## Public Static Functions

QString **getGroupDescription** (int *g*)  
 Low-level function which gets the description text of a group.

```
class FileViewViewmode : public sh::settings::Setting
#include <fileviewviewmode.h>
```

## Public Functions

**FileViewViewmode** ()

QString **name** ()  
 Gets the internal name.

QString **description** ()  
 Gets the description text.

*sh::settings::SettingGroup* **group** ()  
 Gets the group;.

bool **isAdvancedSetting** ()  
 Is this an advanced setting?

void **setValue** (*sh::ui::FileView \*fileview*, QString *value*)  
 Called from Shallot core when the value was set (for a per-fileview setting).

QString **getValue** (*sh::ui::FileView \*filelist*)  
 Get the currently set value.

bool **isGlobal** ()  
 Does this setting apply globally or just for a certain subtree of nodes?

bool **isPerFileview** ()  
 Does this setting apply for each fileview individually or for the complete main window?

QString **valueDescription** (QString *value*)  
 Gets a human readable description text for a value.

void **setValue** (QString)  
 Called from Shallot core when the value was set (for a not-per-fileview setting).

## Public Static Functions

QString **getGroupDescription** (int *g*)  
 Low-level function which gets the description text of a group.

```
class JumpbarMode : public sh::settings::Setting
#include <jumpbarmode.h>
```

## Public Functions

**JumpbarMode** ()

QString **name** ()

Gets the internal name.

QString **description** ()

Gets the description text.

*sh::settings::SettingGroup* **group** ()

Gets the group;.

bool **isAdvancedSetting** ()

Is this an advanced setting?

void **setValue** (QString *value*)

Called from Shallot core when the value was set (for a not-per-fileview setting).

QString **getValue** (*sh::ui::FileView* \**filelist*)

Get the currently set value.

bool **isGlobal** ()

Does this setting apply globally or just for a certain subtree of nodes?

bool **isPerFileview** ()

Does this setting apply for each fileview individually or for the complete main window?

QString **valueDescription** (QString *value*)

Gets a human readable description text for a value.

void **setValue** (*sh::ui::FileView*\*, QString)

Called from Shallot core when the value was set (for a per-fileview setting).

## Public Static Functions

QString **getGroupDescription** (int *g*)

Low-level function which gets the description text of a group.

```
class NumberOfFilePanels : public sh::settings::Setting  
#include <numberoffilepanels.h>
```

## Public Functions

**NumberOfFilePanels** ()

QString **name** ()

Gets the internal name.

QString **description** ()

Gets the description text.

*sh::settings::SettingGroup* **group** ()

Gets the group;.

bool **isAdvancedSetting** ()

Is this an advanced setting?

void **setValue** (QString *value*)

Called from Shallot core when the value was set (for a not-per-fileview setting).



QString **getValue** (*sh::ui::FileView \*filelist*)

Get the currently set value.

bool **isGlobal** ()

Does this setting apply globally or just for a certain subtree of nodes?

bool **isPerFileview** ()

Does this setting apply for each fileview individually or for the complete main window?

QString **valueDescription** (QString *value*)

Gets a human readable description text for a value.

void **setValue** (*sh::ui::FileView\**, QString)

Called from Shallot core when the value was set (for a per-fileview setting).

## Public Static Functions

QString **getGroupDescription** (int *g*)

Low-level function which gets the description text of a group.

```
class ShowHiddenFiles: public sh::settings::Setting
#include <showhiddenfiles.h>
```

## Public Functions

**ShowHiddenFiles** ()

QString **name** ()

Gets the internal name.

QString **description** ()

Gets the description text.

*sh::settings::SettingGroup* **group** ()

Gets the group;.

bool **isAdvancedSetting** ()

Is this an advanced setting?

void **setValue** (*sh::ui::FileView \*filelist*, QString *value*)

Called from Shallot core when the value was set (for a per-fileview setting).

QString **getValue** (*sh::ui::FileView \*filelist*)

Get the currently set value.

bool **isGlobal** ()

Does this setting apply globally or just for a certain subtree of nodes?

bool **isPerFileview** ()

Does this setting apply for each fileview individually or for the complete main window?

QString **valueDescription** (QString *value*)

Gets a human readable description text for a value.

void **setValue** (QString)

Called from Shallot core when the value was set (for a not-per-fileview setting).

### Public Static Functions

QString **getGroupDescription** (int *g*)  
Low-level function which gets the description text of a group.

```
class ShowTree : public sh::settings::Setting
#include <showtree.h>
```

### Public Functions

**ShowTree** ()

QString **name** ()  
Gets the internal name.

QString **description** ()  
Gets the description text.

sh::settings::SettingGroup **group** ()  
Gets the group;.

bool **isAdvancedSetting** ()  
Is this an advanced setting?

void **setValue** (QString *value*)  
Called from Shallot core when the value was set (for a not-per-fileview setting).

QString **getValue** (sh::ui::FileView \**filelist*)  
Get the currently set value.

bool **isGlobal** ()  
Does this setting apply globally or just for a certain subtree of nodes?

bool **isPerFileview** ()  
Does this setting apply for each fileview individually or for the complete main window?

QString **valueDescription** (QString *value*)  
Gets a human readable description text for a value.

void **setValue** (sh::ui::FileView\*, QString)  
Called from Shallot core when the value was set (for a per-fileview setting).

### Public Static Functions

QString **getGroupDescription** (int *g*)  
Low-level function which gets the description text of a group.

```
class SizeFormattingMode : public sh::settings::Setting
#include <sizeformattingmode.h>
```

## Public Functions

**SizeFormattingMode** ()

QString **name** ()

Gets the internal name.

QString **description** ()

Gets the description text.

*sh::settings::SettingGroup* **group** ()

Gets the group;.

bool **isAdvancedSetting** ()

Is this an advanced setting?

void **setValue** (*sh::ui::FileView* \*filelist, QString value)

Called from Shallot core when the value was set (for a per-fileview setting).

QString **getValue** (*sh::ui::FileView* \*filelist)

Get the currently set value.

bool **isGlobal** ()

Does this setting apply globally or just for a certain subtree of nodes?

bool **isPerFileview** ()

Does this setting apply for each fileview individually or for the complete main window?

QString **valueDescription** (QString value)

Gets a human readable description text for a value.

void **setValue** (QString)

Called from Shallot core when the value was set (for a not-per-fileview setting).

## Public Static Functions

QString **getGroupDescription** (int g)

Low-level function which gets the description text of a group.

```
class StickyTreeview: public sh::settings::Setting
#include <stickytreeview.h>
```

## Public Functions

**StickyTreeview** ()

QString **name** ()

Gets the internal name.

QString **description** ()

Gets the description text.

*sh::settings::SettingGroup* **group** ()

Gets the group;.

bool **isAdvancedSetting** ()

Is this an advanced setting?

void **setValue** (QString value)

Called from Shallot core when the value was set (for a not-per-fileview setting).

QString **getValue** (*sh::ui::FileView \*filelist*)

Get the currently set value.

bool **isGlobal** ()

Does this setting apply globally or just for a certain subtree of nodes?

bool **isPerFileview** ()

Does this setting apply for each fileview individually or for the complete main window?

QString **valueDescription** (QString *value*)

Gets a human readable description text for a value.

void **setValue** (*sh::ui::FileView\**, QString)

Called from Shallot core when the value was set (for a per-fileview setting).

## Public Static Functions

QString **getGroupDescription** (int *g*)

Low-level function which gets the description text of a group.

```
class WindowTitle : public sh::settings::Setting
#include <windowtitle.h>
```

## Public Functions

**WindowTitle** ()

QString **name** ()

Gets the internal name.

QString **description** ()

Gets the description text.

*sh::settings::SettingGroup* **group** ()

Gets the group;.

bool **isAdvancedSetting** ()

Is this an advanced setting?

void **setValue** (QString *value*)

Called from Shallot core when the value was set (for a not-per-fileview setting).

QString **getValue** (*sh::ui::FileView \*filelist*)

Get the currently set value.

bool **isGlobal** ()

Does this setting apply globally or just for a certain subtree of nodes?

bool **isPerFileview** ()

Does this setting apply for each fileview individually or for the complete main window?

void **setValue** (*sh::ui::FileView\**, QString)

Called from Shallot core when the value was set (for a per-fileview setting).

QString **valueDescription** (QString *value*)

Gets a human readable description text for a value.

## Public Static Functions

QString **getGroupDescription** (int g)

Low-level function which gets the description text of a group.

**namespace tools**

Auxiliary stuff.

**class AtomicCounter**

*#include <atomiccounter.h>* Internal tool for reference counting.

## Public Functions

**AtomicCounter** ()

int **value** ()

std::shared\_ptr<*AtomicCounter::Increment*> **increment** ()

int **doIncValue** ()

int **doDecValue** ()

## Private Members

int **\_value** = 0

QMutex **\_mutex**

**class Increment**

*#include <atomiccounter.h>*

## Public Functions

**Increment** (*AtomicCounter* \*ac)

**~Increment** ()

## Private Members

*AtomicCounter* \*\_ac

## Friends

**friend class** AtomicCounter

**class Benchmarking** : **public** QObject, **public** *sh::base::Singleton*

*#include <benchmarking.h>* Tools for performance measurements.

## Public Types

**typedef** QPair<QString, qint64> **BenchmarkFrame**  
A pair of a benchmark keyname and a duration value.

## Public Functions

**Benchmarking** ()

void **shutdown** ()  
Shutdown down this singleton.

bool **isAlive** ()  
Returns if this singleton is alive (true until its shutdown begins).

**class** **Bookmark**  
*#include <bookmarkmanager.h>* A bookmark.

## Public Functions

**Bookmark** (QString *id*, QStringList *folder*, QString *label*, std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*, QString *tags*)  
Constructed only by the infrastructure and made available otherwise.

QString **id** ()  
The bookmark id (used in some methods of *sh::tools::BookmarkManager*).

QStringList **folder** ()  
The bookmark folder (the user interface calls it ‘Collections’).

QString **label** ()  
The bookmark label.

std::shared\_ptr<const *sh::filesystem::Eurl*> **eurl** ()  
The location this bookmark points to.

QString **tags** ()  
Internal information for bookkeeping.  
Can be used by external code for keeping track of dynamically created bookmarks.

## Private Members

QString **\_id**

QStringList **\_folder**

QString **\_label**

std::shared\_ptr<const *sh::filesystem::Eurl*> **\_eurl**

QString **\_tags**

**class** **BookmarkManager** : public QObject, public *sh::base::Singleton*  
*#include <bookmarkmanager.h>* The bookmark manager.  
Use it for getting or modifying bookmarks.

## Public Functions

`QList<std::shared_ptr<sh::tools::Bookmark>> getBookmarks ()`

Returns a list of all stored bookmarks.

`bool hasBookmarks ()`

Checks if any bookmarks exist.

`QString addBookmark (QList<QString> folder, QString label, std::shared_ptr<const sh::filesystem::Eurl> eurl, QString tags = QString())`

Adds a new bookmark.

**Return** The id of the new bookmark.

`void removeBookmark (QString id)`

Removes a bookmarks.

`void changeBookmark (QString id, QString label, std::shared_ptr<const sh::filesystem::Eurl> eurl)`

Changes data of a bookmark.

`void changeBookmarkTags (QString id, QString tags)`

Changes the tags of a bookmark.

`void moveBookmarkUp (QString id)`

Moves a bookmark up within its folder.

`void moveBookmarkDown (QString id)`

Moves a bookmark down within its folder.

`void moveBookmarkToFolder (QString id, QList<QString> folder)`

Changes to folder of a bookmark.

`void doInitialize ()`

Executes singleton initialization.

`void doShutdown ()`

Executes singleton shutdown.

`void shutdown ()`

Shutdown down this singleton.

`bool isAlive ()`

Returns if this singleton is alive (true until its shutdown begins).

## Signals

`void changed ()`

## Private Functions

`BookmarkManager ()`

`void moveBookmark (QString id, int direction)`

`void _changeBookmark (QString id, std::function<QString> std::shared_ptr<Bookmark>  
> label, std::function<std::shared_ptr<const sh::filesystem::Eurl>std::shared_ptr<Bookmark>>  
eurl, std::function<QList<QString>std::shared_ptr<Bookmark>> folder,  
std::function<QStringstd::shared_ptr<Bookmark>> tags`

`void readBookmarks ()`

```
void writeBookmarks ()
```

## Private Members

```
std::shared_ptr<sh::configuration::ConfigurationValue> cfgvalBookmarks
```

```
QList<std::shared_ptr<Bookmark>> _bookmarks
```

```
QMutex _bookmarksmutex
```

```
class DataExchange : public QObject, public sh::base::Singleton
#include <dataexchange.h> Clipboard and DnD related tools.
```

## Public Types

```
enum DataExchangeType
```

A kind of data exchange movement.

Values:

```
enumerator COPY
```

```
enumerator MOVE
```

## Public Functions

```
bool containsFileEntries (const QMimeData*)
```

Checks if a QMimeData contains any file entries we understand.

```
QMimeData *getMimeDataFromFilesystemNodes (QList<std::shared_ptr<sh::filesystem::FilesystemNode>>,
                                             DataExchangeType type)
```

Returns a new QMimeData for a list of filesystem nodes and an exchange type.

```
std::shared_ptr<sh::actions::AbstractActionItem> createCopyAction (const QMime-
                                                                    Data      *src,
                                                                    std::shared_ptr<const
sh::filesystem::Eurl>
                                                                    dest)
```

Returns a copy action for a source by QMimeData and a destination eurl.

```
std::shared_ptr<sh::actions::AbstractActionItem> createMoveAction (const QMime-
                                                                    Data      *src,
                                                                    std::shared_ptr<const
sh::filesystem::Eurl>
                                                                    dest)
```

Returns a move action for a source by QMimeData and a destination eurl.

```
std::shared_ptr<sh::actions::AbstractActionItem> createPasteAction (const QMime-
                                                                    Data      *src,
                                                                    std::shared_ptr<const
sh::filesystem::Eurl>
                                                                    dest)
```

Returns a paste action for a source by QMimeData and a destination eurl (copy or move by QMimeData).

```
QList<std::shared_ptr<const sh::filesystem::Eurl>> getSources (const QMimeData *src,
                                                                bool *isCut = 0)
```

Returns a list of eurls (and if it is a cut/move exchange) from the QMimeData.



```
void doInitialize ()
    Executes singleton initialization.

void doShutdown ()
    Executes singleton shutdown.

void shutdown ()
    Shutdown down this singleton.

bool isAlive ()
    Returns if this singleton is alive (true until its shutdown begins).
```

### Public Static Attributes

```
QString FilelistTypeShallot = "x-special/shallot-copied-files"
QString FilelistTypeGnome = "x-special/gnome-copied-files"
QString FilelistTypeUrulist = "text/uri-list"
QString FilelistTypePlaintext = "text/plain"
```

### Private Functions

```
DataExchange ()
```

### Private Static Attributes

```
QString linebreak = "\n"
```

```
template<class T>
```

```
class HistoryTracker
```

```
    #include <historytracker.h> A data structure for tracking a history of data.
```

It is a templated container It is typically used for tracking the directory history (used by *sh::actions::common::ActionHistoryNavigate*, et al).

### Public Functions

```
HistoryTracker ()
```

```
void visitValue (T v)
```

```
int count ()
```

```
QList<HistoryEntry> forwardList ()
```

```
QList<HistoryEntry> backwardList ()
```

```
void revisitValue (int idx)
```

### Public Members

```
const int HISTORY_SIZE = 10
```

### Private Members

```
QList<T> _items
```

```
int _current = -1
```

```
class HistoryEntry
    #include <historytracker.h>
```

### Public Functions

```
HistoryEntry (int i, T v)
```

```
int index ()
```

```
T value ()
```

### Private Members

```
int _index
```

```
T _value
```

```
class Jsonable
```

*#include <misc.h>* An interface for objects which can return a json representation as QJsonObject.

Subclassed by *sh::ui::web::WebDialog*, *sh::ui::web::WebFilePropertyDialog::WebFilePropertyDialogTabActionsView*,  
*sh::ui::web::WebFilePropertyDialog::WebFilePropertyDialogTabIconTextBannerView*,  
*sh::ui::web::WebFilePropertyDialog::WebFilePropertyDialogTabTableView*,  
*sh::ui::web::WebFilePropertyDialog::WebFilePropertyDialogTabTextView*

### Public Functions

```
QJsonObject toJson () = 0
```

Returns the json representation as QJsonObject.

```
class LocalFile : public QFile
```

*#include <datastream.h>* A QFile (and a QIODevice) for local file access.

It also stores the information whether this is a temporary location or the permanent one.

### Public Functions

```
LocalFile (QString &file, bool isTemp = false)
```

```
bool isTemp ()
```

## Private Members

bool **\_istemp**

**class LocalFilesystemWatcher** : public QObject, public *sh::base::Singleton*  
*#include <localfilesystemwatcher.h>* Used for watching parts of the local filesystem.

Depending on the compile flags and your system, this functionality might not be available. If so, the methods are no-ops.

## Public Functions

qint64 **addFile** (QString *path*)

Watches one more file.

See also *removeFile()*.

void **removeFile** (qint64 *id*)

Stops watching a file (by the return value of *addFile()*).

qint64 **addDirectory** (QString *path*)

Watches one more directory.

See also *removeDirectory()*.

void **removeDirectory** (qint64 *id*)

Stops watching a directory (by the return value of *addDirectory()*).

**~LocalFilesystemWatcher** ()

void **doInitialize** ()

Executes singleton initialization.

void **doShutdown** ()

Executes singleton shutdown.

void **shutdown** ()

Shutdown down this singleton.

bool **isAlive** ()

Returns if this singleton is alive (true until its shutdown begins).

## Signals

void **elementModified** (QString *name*, QList<qint64> *ids*)

void **elementDeleted** (QString *name*, QList<qint64> *ids*)

void **elementCreated** (QString *name*, QList<qint64> *ids*)

### Private Functions

```
LocalFilesystemWatcher ()  
void emitElementModified (QString name, QList<qint64> ids)  
void emitElementDeleted (QString name, QList<qint64> ids)  
void emitElementCreated (QString name, QList<qint64> ids)
```

### Private Members

```
InotifyThread *inotifyThread  
QMutex inotifymutex  
QHash<int, QList<qint64>> inotifywd2ids  
QHash<qint64, int> id2inotifywd  
QHash<qint64, QString> id2path  
qint64 nextEid = 0  
QMutex inotifythreadmutex  
QWaitCondition inotifythreadmutexwait
```

### Friends

```
friend class InotifyThread  
class InotifyThread : public QThread
```

### Public Functions

```
InotifyThread (LocalFilesystemWatcher *watcher)
```

### Friends

```
friend class LocalFilesystemWatcher  
class LocalFilesystemWatcherConnector : public QObject, public sh::base::Singleton  
#include <localfilesystemwatcherconnector.h> Observes the list of file views via VisibleViews and  
controls LocalFilesystemWatcher with that for observing the filesystem for changes.
```

### Public Functions

```
void doInitialize ()  
    Executes singleton initialization.  
void doShutdown ()  
    Executes singleton shutdown.  
void shutdown ()  
    Shutdown down this singleton.
```

bool **isAlive** ()

Returns if this singleton is alive (true until its shutdown begins).

## Private Functions

**LocalFilesystemWatcherConnector** ()

QList<QPair<std::shared\_ptr<*sh::filesystem::FilesystemNode*>, *sh::filesystemhandlers::LocalFilesystemHandler*\*>> **ge**

## Private Members

QHash<std::shared\_ptr<const *sh::filesystem::Eurl*>, QList<qint64>> **dir2watcherids**

QHash<qint64, std::shared\_ptr<const *sh::filesystem::Eurl*>> **watcherid2dir**

**class Misc**

*#include <misc.h>*

## Public Static Functions

QByteArray **iconToPngByteArray** (QIcon *icon*, int *sizeInPt*)

Computes a png representation for an icon.

QByteArray **iconToBase64SrcEncoding** (QIcon *icon*, int *sizeInPt*)

Computes a html-compatible 'data:image/png;base64,...' png representation for an icon.

QByteArray **hash** (QStringList *data*)

Hashes string data in a cryptographical way (with a salt).

Use with *compareHash()*.

QByteArray **hashUnsalted** (QStringList *data*)

Hashes string data in a cryptographical way without a salt.

Use with *compareHash()* or just compare strings.

bool **compareHash** (QByteArray *hash*, QStringList *data*)

Check if a hash is matching to given string data.

QByteArray **qjsonToJson** (QJsonObject *o*)

Returns a json string for a QJsonObject, QJsonValue or QJsonArray.

QByteArray **qjsonToJson** (QJsonValue *o*)

QByteArray **qjsonToJson** (QJsonArray *a*)

QByteArray **qmapToJson** (QVariantMap *m*)

Returns a json string for a QVariantMap.

QByteArray **jsonableToJson** (*Jsonable* \**a*)

Returns a json string from a *Jsonable*.

QByteArray **randomBytes** (int *length* = 32)

Returns a random byte array.

QByteArray **generateUniqueHash** ()

Returns a (mostly) random generated unique hash.

void **makeHttpRequest** (QUrl *url*, QByteArray *\*responseBody* = nullptr, QString *\*mimeType* = nullptr, int *\*httpStatus* = nullptr)  
Makes an http request (as a client).  
*httpStatus* can return 0 for network errors and similar.

## Private Functions

**Misc** ()

## Private Static Functions

QByteArray **\_hash** (QStringList *data*, bool *salted*, QByteArray *\_withSalt*)

## Private Static Attributes

QByteArray **\_jundefined**

QNetworkAccessManager **\_qnetwork**

**class OperationsCache** : public QObject, public *sh::base::Singleton*  
*#include <operationscache.h>* A cached factory for readonly *sh::filesystem::Operation* instances.

You can use it for fetching an *sh::filesystem::Operation* for a certain *sh::filesystem::Eurl*. Two restrictions apply:

- Data might be slightly outdated.
- Just for read-only access.

## Public Functions

std::shared\_ptr<*sh::filesystem::Operation*> **getOperationForContainer** (std::shared\_ptr<const *sh::filesystem::Eurl*> *container*)

Returns a cached *sh::filesystem::Operation* for a container (by eurl).

void **doInitialize** ()  
Executes singleton initialization.

void **doShutdown** ()  
Executes singleton shutdown.

void **shutdown** ()  
Shutdown down this singleton.

bool **isAlive** ()  
Returns if this singleton is alive (true until its shutdown begins).

## Private Functions

**OperationsCache** ()

## Private Members

QHash<std::shared\_ptr<const *sh::filesystem::Eurl*>, std::shared\_ptr<*sh::filesystem::Operation*>> **cache**  
 QMutex **cachemutex**

**class ReadDataDevice** : public QIODevice  
*#include <datastream.h>* A QIODevice implementation, acting as an abstract base class for other subclasses.

The provided interface is not as generic but easier than the original one for many easier tasks.

Subclassed by *sh::scripting::api::ApiReadDataDevice*

## Public Functions

**ReadDataDevice** ()

QByteArray **getdata** () = 0

This method returns the next available chunk of data. It may return empty arrays whenever temporarily no data is available. Returning a null array (with `QByteArray::isNull() == true`) marks the end of the stream. .

## Private Members

QByteArray **current**

int **currentlen** = 0

int **currentconsumed** = 0

**class ThumbnailManager** : public QObject, public *sh::base::Singleton*  
*#include <thumbnailmanager.h>* Creates thumbnail images for filesystem nodes.

This is a cached source.

## Public Functions

void **requestThumbnail** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> *node*, int *width*, int *height*, std::function<void() QIcon  
 > *callback* = 0) Requests a thumbnail in a given size for a given node.

### Parameters

- *callback*: Called when the thumbnail is ready.

bool **getThumbnail** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> *node*, int *width*, int *height*, QIcon \**result*, bool \**outdated* = 0, bool \**refreshRequested* = 0)  
 Returns a thumbnail in a given size for a given node from the current cache.

void **invalidateCache** ()  
 Invalidates the thumbnail cache.

void **addThumbnailProvider** (int *index*, std::shared\_ptr<*ThumbnailProvider*> *provider*)  
 Adds a thumbnail provider.

void **doInitialize** ()  
    Executes singleton initialization.

void **shutdown** ()  
    Shutdown down this singleton.

bool **isAlive** ()  
    Returns if this singleton is alive (true until its shutdown begins).

## Signals

void **thumbnailAvailable** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> node)  
    Emitted when a new thumbnail is available.

## Public Static Attributes

const int **REGISTER\_THUMBNAILPROVIDER\_INDEX\_CORE** = 1000000  
    Base value for display indexes of core (i.e. very important) thumbnail providers.  
    Used in ThumbnailProvider::addThumbnailProvider.

const int **REGISTER\_THUMBNAILPROVIDER\_INDEX\_NORMAL** = 2000000  
    Base value for display indexes of thumbnail providers with normal importance.  
    Used in ThumbnailProvider::addThumbnailProvider.

const int **REGISTER\_THUMBNAILPROVIDER\_INDEX\_EXOTIC** = 3000000  
    Base value for display indexes of exotic thumbnail providers.  
    Used in ThumbnailProvider::addThumbnailProvider.

const int **REGISTER\_THUMBNAILPROVIDER\_INDEX\_FALLBACK** = 4000000  
    Base value for display indexes of fallback thumbnail providers (those who try to get something when everything else failed).  
    Used in ThumbnailProvider::addThumbnailProvider.

## Private Functions

**ThumbnailManager** ()

void **worker** ()

void **\_enforce\_capacity** ()

## Private Members

QHash<*sh::filesystem::FilesystemNode*\*, *Thumbnail*\*> **thumbnails**

QMutex **thumbnailslock**

QList<*ThumbnailRequest*> **requestQueue**

int **maxworkers**

int **runningworkers**

QMutex **workermutex**

int **capacity**



```

qint64 _curr_accessTime = 0
QHash<int, std::shared_ptr<ThumbnailProvider>> _thumbnailProviderMap
QList<std::shared_ptr<ThumbnailProvider>> _thumbnailProviders
QMutex _thumbnailProvidersMutex
class Thumbnail

```

### Public Functions

```

Thumbnail (QIcon icon, qint64 validUntil, qint64 accessTime,
           std::weak_ptr<sh::filesystem::FilesystemNode> node, int width, int height)

```

### Public Members

```

QIcon icon
qint64 validUntil
qint64 accessTime
bool refreshRequested
std::weak_ptr<sh::filesystem::FilesystemNode> node
int width
int height
class ThumbnailRequest

```

### Public Functions

```

ThumbnailRequest (std::weak_ptr<sh::filesystem::FilesystemNode> node, int width, int
                  height, QList<std::function<void>()> QIcon
                  >> callbacks
bool operator== (ThumbnailRequest const &b)

```

### Public Members

```

std::weak_ptr<sh::filesystem::FilesystemNode> node
int width
int height
QList<std::function<void (QIcon)>> callbacks
class ThumbnailSortStruct

```

## Public Functions

**ThumbnailSortStruct** (*sh::filesystem::FileSystemNode \*node*, quint64 *accessTime*)

## Public Members

*sh::filesystem::FileSystemNode \*node*

quint64 **accessTime**

**class ThumbnailProvider**

*#include <thumbnailmanager.h>* Subclassed by *sh::scripting::api::ApiThumbnailProvider*,  
*sh::tools::thumbnailproviders::DefaultImageThumbnailProvider*, *sh::tools::thumbnailproviders::FfmpegVideoThumbn*  
*sh::tools::thumbnailproviders::ImageMagickPdfThumbnailProvider*,  
*sh::tools::thumbnailproviders::PlaintextThumbnailProvider*

## Public Functions

void **getThumbnail** (*sh::filesystem::Operation \*operation*, std::shared\_ptr<*sh::filesystem::FileSystemNode*>  
*node*, QString *contentType*, int *width*, int *height*, QIcon *icon*) = 0

**class UserDirLock**

*#include <userdirlock.h>* Locks the Shallot user directory during exclusive usage.

This lock is rather heavy-weight and should be used only if necessary!

## Public Functions

**UserDirLock** ()

Is intended to be directly constructed from everywhere.

**~UserDirLock** ()

## Private Members

bool **dounlock**

QString **slockpref**

QString **slockfile**

## Private Static Attributes

QThread **\*currentThread** = 0

QMutex **mutex**

**class VisibleViews** : public *sh::base::Singleton*

*#include <visibleviews.h>* Used for keeping track of which fileviews show which directories.

Whenever the view goes to other directories, this manager gets notified and redirects the notification to some other parts of the program.

## Public Functions

`std::shared_ptr<QObject> viewEnteredDirectory (std::shared_ptr<sh::filesystem::FilesystemNode>  
dir)  
 Called when a view enters the given directory node. .`

`void registerOnEnteredHandlers (std::function<void> std::shared_ptr<sh::filesystem::FilesystemNode>  
dir  
> handler)Registers a function to be called whenever the view just entered the given directory  
 node.`

`void registerOnLeftHandlers (std::function<void> std::shared_ptr<sh::filesystem::FilesystemNode>  
dir  
> handler)Registers a function to be called whenever the view just left the given directory node.`

`void doInitialize ()`  
 Executes singleton initialization.

`void doShutdown ()`  
 Executes singleton shutdown.

`void shutdown ()`  
 Shutdown down this singleton.

`bool isAlive ()`  
 Returns if this singleton is alive (true until its shutdown begins).

## Private Functions

`VisibleViews ()`

## Private Members

`QHash<std::shared_ptr<sh::filesystem::FilesystemNode>, int> visibleDirs`

`QList<std::function<void (std::shared_ptr<sh::filesystem::FilesystemNode>) >>  
onenteredhandlers`

`QList<std::function<void (std::shared_ptr<sh::filesystem::FilesystemNode>) >>  
onlefthandlers`

### **namespace accounts**

*Account* and password storage.

See *sh::tools::accounts::AccountsManager* for more.

### **class AbstractAccountsProvider**

*#include <abstractaccountsprovider.h>* Abstract base class for accounts provider.

Implement this class (and register an instance of it) in order to add support for something like a password/account manager.

Subclassed by *sh::tools::accounts::FallbackAccountsProvider*,  
*sh::tools::accounts::LibsecretAccountsProvider*

### Public Functions

**AbstractAccountsProvider** ()

**~AbstractAccountsProvider** ()

void **findAccounts** (std::shared\_ptr<*Account*> *pattern*, QList<std::shared\_ptr<*Account*>> &*result*) = 0

Finds account info by a given pattern.

bool **storeAccount** (std::shared\_ptr<*Account*> *account*) = 0

Stores account infos.

**class Account**

*#include <accountsmanager.h>* *Account* data (for accessing network drives).

### Public Functions

**Account** () = default

**Account** (const *Account* &*other*) = default

### Public Members

QString **username**

QString **domain**

QString **server**

int **serverport** = -1

QString **path**

QString **protocol**

QString **authtype**

QByteArray **authinfo**

**class AccountsManager** : public *sh::base::Singleton*

*#include <accountsmanager.h>* Storage for account data (for accessing network drives).

### Public Functions

QList<std::shared\_ptr<*Account*>> **findAccounts** (std::shared\_ptr<*Account*> *pattern*)

Finds account infos by a given pattern.

void **storeAccount** (std::shared\_ptr<*Account*> *account*)

Stores account infos.

void **registerProvider** (std::shared\_ptr<*AbstractAccountsProvider*> *provider*)

Registers a new accounts provider.

void **doInitialize** ()

Executes singleton initialization.

void **doShutdown** ()

Executes singleton shutdown.

```
void shutdown ()
    Shutdown down this singleton.

bool isAlive ()
    Returns if this singleton is alive (true until its shutdown begins).
```

### Private Functions

```
AccountsManager ()
```

### Private Members

```
QList<std::shared_ptr<AbstractAccountsProvider>> accountsproviders

QMutex mutex
```

```
class FallbackAccountsProvider : public sh::tools::accounts::AbstractAccountsProvider
    #include <fallbackaccountsprovider.h> A fallback accounts provider which at least stores user
    names (i.e. no passwords) locally on disk.
```

### Public Functions

```
FallbackAccountsProvider ()

void findAccounts (std::shared_ptr<Account> pattern, QList<std::shared_ptr<Account>>
                    &result)
    Finds account info by a given pattern.

bool storeAccount (std::shared_ptr<Account> account)
    Stores account infos.
```

### Public Static Functions

```
void doInitialize ()

void doShutdown ()
```

### Private Functions

```
QList<std::shared_ptr<Account>> _find_rem_account_helper (std::shared_ptr<Account>
                                                         pattern, bool re-
                                                         move)
```

### Private Members

```
QString accountsdir
```

```
class LibsecretAccountsProvider : public sh::tools::accounts::AbstractAccountsProvider
    #include <libsecretaccountsprovider.h> An accounts provider based on gnome-keyring.
```

## Public Functions

## LibsecretAccountsProvider ()

```
void findAccounts (std::shared_ptr<Account> pattern, QList<std::shared_ptr<Account>>
                    &result)
```

Finds account info by a given pattern.

```
bool storeAccount (std::shared_ptr<Account> account)
```

### Stores account infos.

## Public Static Functions

```
void doInitialize ()
```

```
void doShutdown ( )
```

## Private Members

```
QByteArray sauthtype = QString("authtype").toUtf8()
```

```
ByteArray sdomain = OString("domain").toUtf8()
```

```
QByteArray spath = QString("object").toUtf8()
```

```
QByteArray sprotocol = QString("protocol").toUtf8()
```

```
QByteArray sserver = QString("server").toUtf8()
```

```
QByteArray sserverport = QString("port").toUtf8()
```

```
QByteArray susername = QString("user").toUtf8()
```

```
namespace filetypes
```

Tools for determining a file's type (pdf, image, mp3, et al) and ways how to deal with that.

See *sh::tools::filetypes::FileTypeManager* for more.

```
class FileTypeManager : public QObject, public sh::base::Singleton
```

```
#include <filetypemanager.h> Utilities for dealing with file types.
```

It can determine the type of a file (png, plaintext, html, . . . ), it can provide information about how to open them with an external program, and more.

For most tasks it uses a pluggable interface. Actual implementations of strategies for those tasks reside in separate classes in this namespace.

## Public Functions

```
QString determineMimetype (sh::filesystem::Operation *op,    std::shared_ptr<const  
                           sh::filesystem::Eurl> eurl)
```

Determines the mimetype for one file.

```
QHash<std::shared_ptr<const sh::filesystem::Eurl>, QString> determineMimetype(sh::filesystem::Operation  
    *op,  
    QList<std::shared_ptr<  
        sh::filesystem::Eurl>>  
    items)
```

Determines the mimetypes for a list of files.

`QList<std::shared_ptr<OpenMethod>> getOpenMethods (QString mimetype,  
QList<std::shared_ptr<sh::filesystem::FilesystemNodes>>  
nodes)`

Determines how to open a file with a given mimetype with an external program.

`QList<std::shared_ptr<OpenMethod>> getAllOpenMethods ()`

Returns a list of all known infos how to open a file external programs.

This has roughly one entry for each installed program on the user's system, which can graphically open files.

`void doInitialize ()`

Executes singleton initialization.

`void doShutdown ()`

Executes singleton shutdown.

`void shutdown ()`

Shutdown down this singleton.

`bool isAlive ()`

Returns if this singleton is alive (true until its shutdown begins).

## Private Functions

`FileTypeManager (QObject *parent = 0)`

## Private Members

`QList<std::shared_ptr<MimetypeDeterminationStrategy>> _mimetypeDeterminationMethods`

`QList<std::shared_ptr<OpenMethodDeterminationStrategy>> _openMethodDeterminationMethods`

`QList<std::shared_ptr<MimetypeInformationRetrievalStrategy>> _mimetypeInformationRetrievalMethods`

`QMutex _mutex`

`class MimetypeDeterminationStrategy`

`#include <filetypemanager.h> Abstract base class for a mimetype determination strategy.`

Subclassed by `sh::tools::filetypes::FreedesktopOrgToolsMimetypeDeterminationStrategy`,

`sh::tools::filetypes::SuffixListMimetypeDeterminationStrategy`,

`sh::tools::filetypes::UnixFileToolMimetypeDeterminationStrategy`

## Public Functions

`QHash<std::shared_ptr<const sh::filesystem::Eurl>, QString> determineMimetype (sh::filesystem::Operation  
*op,  
QList<std::shared_ptr<sh::filesystem::Eurl>>  
items)  
=  
0`

`~MimetypeDeterminationStrategy ()`

`class MimetypeInformationRetrievalStrategy`

`#include <filetypemanager.h> Abstract base class for mimetype information retrieval strategy.`

Subclassed by `sh::tools::filetypes::FreeDesktopOrgMimetypeInformationRetrievalStrategy`

### Public Functions

```
QStringList getMimeTypeSubclasses (QString mimetype)  
~MimeTypeInformationRetrievalStrategy ()  
class OpenMethodDeterminationStrategy  
    #include <filetypemanager.h> Abstract base class for a ‘open method’ determination strategy.  
    Subclassed by sh::tools::filetypes::FreedesktopOrgToolsOpenMethodDeterminationStrategy,  
    sh::tools::filetypes::UserDefinedOpenMethodDeterminationStrategy
```

### Public Functions

```
QList<std::shared_ptr<OpenMethod>> getOpenMethods (QString mimetype,  
                                                    QList<std::shared_ptr<sh::filesystem::FilesystemNode>>  
                                                    nodes) = 0  
~OpenMethodDeterminationStrategy ()  
class FreeDesktopOrgMimeTypeInformationRetrievalStrategy : public sh::tools::filetypes::File  
    #include <freedesktoporgmimetypeinformationretrievalstrategy.h> Tries to determine some  
    mimetype information with the freedesktop.org specs.
```

### Public Functions

```
FreeDesktopOrgMimeTypeInformationRetrievalStrategy ()  
QStringList getMimeTypeSubclasses (QString mimetype)
```

### Private Members

```
QHash<QString, QStringList> _mimeSubclassOf  
QMutex _mutex  
QWaitCondition _cond_initied  
bool _initied = false  
class MimeTypeInfo : public QXmlDefaultHandler
```

### Public Functions

```
MimeTypeInfo () = default  
bool startElement (const QString &namespaceURI, const QString &localName,  
                  const QString &qName, const QXmlAttributes &atts)  
bool endElement (const QString &namespaceURI, const QString &localName,  
                 const QString &qName)
```



## Public Members

QStringList **subClassOf**

**class FreedesktopOrgToolsMimetypeDeterminationStrategy** : public *sh::tools::filetypes::FileTypeManager*  
*#include <freedesktoporgtoolsmimetypedeterminationmethod.h>* Tries to determine a file's  
 mimetype with the freedesktop.org tools.

## Public Functions

**FreedesktopOrgToolsMimetypeDeterminationStrategy** (*sh::tools::filetypes::FileTypeManager*  
*\*manager*)

QHash<std::shared\_ptr<const *sh::filesystem::Eurl*>, QString> **determineMimetype** (*sh::filesystem::Operation*  
*\*op*,  
 QList<std::shared\_ptr<*sh::filesystem::Eurl*>>  
*items*)

## Private Members

QMutex **\_mutex**

QString **\_pathToFileTool**

const QRegularExpression **\_reMimetype**

## Private Static Attributes

std::shared\_ptr<*sh::configuration::ConfigurationValue*> **cfgvalXdgmimePath** = *sh::configuration::ConfigurationValue*

**class FreedesktopOrgToolsOpenMethodDeterminationStrategy** : public *sh::tools::filetypes::FileTypeManager*  
*#include <freedesktoporgtoolsopenmethoddeterminationmethod.h>* Tries to determine a 'open  
 method' for a file with the freedesktop.org tools.

## Public Functions

**FreedesktopOrgToolsOpenMethodDeterminationStrategy** ()

**~FreedesktopOrgToolsOpenMethodDeterminationStrategy** ()

QList<std::shared\_ptr<*sh::tools::filetypes::OpenMethod*>> **getOpenMethods** (QString  
*mimetype*,  
 QList<std::shared\_ptr<*sh::filesystem::Node*>>  
*nodes*)

### Private Functions

```
QString _parseValue (QStringList content, QString key)  
void _parseExecLine (QString exeline, QString *command, QStringList *arguments)  
QString _escapeExecLineToken (QString s)
```

### Private Members

```
QMultiMap<QString, ApplicationEntry*> _mimetype2applicationEntry  
QMutex _mutex  
QWaitCondition _cond_inited  
bool _inited = false  
struct ApplicationEntry
```

### Public Members

```
QString name  
QStringList mimetypes  
QString command  
QStringList commandargs  
QIcon icon  
bool hidden
```

### **struct OpenMethod**

```
#include <filetypemanager.h> Commandline and infos for opening a file with an external program.
```

### Public Functions

```
OpenMethod (QString name, QString command, QStringList arguments, QIcon icon = QIcon(), int precedence = 0)
```

### Public Members

```
const QString name  
const QString command  
const QStringList arguments  
const QIcon icon  
const int precedence
```

### **class SuffixListMimetypeDeterminationStrategy**: **public** *sh::tools::filetypes::FileTypeManager::M*

```
#include <suffixlistmimetypedeterminationmethod.h> Tries to determine a file's mimetype with an internal lookup table of file extensions.
```

## Public Functions

**SuffixListMimetypeDeterminationStrategy** ()

QHash<std::shared\_ptr<const *sh::filesystem::Eurl*>, QString> **determineMimetype** (*sh::filesystem::Operation*  
\*op,  
QList<std::shared\_ptr<  
*sh::filesystem::Eurl*>>  
items)

## Private Members

QHash<QString, QString> **\_suffixToMimetype**

QMutex **\_mutex**

**class UnixFileToolMimetypeDeterminationStrategy : public *sh::tools::filetypes::FileTypeManager***  
#include <unixfiletoolmimetyperedeterminationmethod.h> Tries to determine a file's mimetype  
with the unix file tool.

## Public Functions

**UnixFileToolMimetypeDeterminationStrategy** (*sh::tools::filetypes::FileTypeManager*  
\*manager)

QHash<std::shared\_ptr<const *sh::filesystem::Eurl*>, QString> **determineMimetype** (*sh::filesystem::Operation*  
\*op,  
QList<std::shared\_ptr<  
*sh::filesystem::Eurl*>>  
items)

## Public Static Attributes

std::shared\_ptr<*sh::configuration::ConfigurationValue*> **cfgvalFilePath** = *sh::configuration::ConfigurationManager*

## Private Members

QString **\_pathToFileTool**

const QRegularExpression **\_reMimetype**

QMutex **\_mutex**

**class UserDefinedOpenMethodDeterminationStrategy : public *sh::tools::filetypes::FileTypeManager***  
#include <userdefinedopenmethoddeterminationstrategy.h> Tries to determine a 'open method'  
for a file by information the user stored before.

## Public Functions

```
QList<std::shared_ptr<sh::tools::filetypes::OpenMethod>> getOpenMethods (QString
                                                                    mimetype,
                                                                    QList<std::shared_ptr<sh::filesystem::
                                                                    nodes>
                                                                    override

void storeCustomOpenMethod (QList<std::shared_ptr<sh::filesystem::FileSystemNode>>
                                                                    nodes,
                                                                    QString
                                                                    mimetype,
                                                                    std::shared_ptr<sh::tools::filetypes::OpenMethod> m,
                                                                    bool rememberForMimetype, std::shared_ptr<const
                                                                    sh::filesystem::Eurl> rememberForDirectory, bool
                                                                    rememberForFile)

    Stores a custom open method.

void doInitialize ()
    Executes singleton initialization.

void doShutdown ()
    Executes singleton shutdown.

void shutdown ()
    Shutdown down this singleton.

bool isAlive ()
    Returns if this singleton is alive (true until its shutdown begins).
```

## Private Functions

```
UserDefinedOpenMethodDeterminationStrategy ()
```

## Private Members

```
QMutex _mutex
```

```
namespace thumbnailproviders
```

Thumbnail providers.

Subclasses of *sh::tools::ThumbnailProvider* (and possibly some auxiliary stuff). Those classes are used for generating thumbnail pictures for files.

```
class DefaultImageThumbnailProvider : public sh::tools::ThumbnailProvider
    #include <defaultimagethumbnailprovider.h> Thumbnail provider for images.
```

## Public Functions

```
DefaultImageThumbnailProvider () = default
```

```
void getThumbnail (sh::filesystem::Operation
                                                                    *operation,
                                                                    std::shared_ptr<sh::filesystem::FileSystemNode> node,
                                                                    QString
                                                                    contentType, int width, int height, QIcon *icon) override
```

## Public Static Functions

void **doInitialize** ()

void **doShutdown** ()

**class FfmpegVideoThumbnailProvider** : public QObject, public *sh::tools::ThumbnailProvider*  
#include <ffmpegvideothumbnailprovider.h> Thumbnail provider for videos using the ffmpeg tool.

## Public Functions

**FfmpegVideoThumbnailProvider** ()

void **getThumbnail** (*sh::filesystem::Operation* \*operation,  
std::shared\_ptr<*sh::filesystem::FilesystemNode*> node, QString  
contentType, int width, int height, QIcon \*icon) **override**

## Public Static Functions

void **doInitialize** ()

void **doShutdown** ()

## Private Members

QString **pathToFfmpegTool**

QString **pathToFfprobeTool**

const QRegularExpression **reDuration**

QMutex **mutexReDuration**

## Private Static Attributes

std::shared\_ptr<*sh::configuration::ConfigurationValue*> **cfgvalFfmpegPath** = *sh::configuration::Configuration*

**class ImageMagickPdfThumbnailProvider** : public QObject, public *sh::tools::ThumbnailProvider*  
#include <imagemagickpdfthumbnailprovider.h> Thumbnail provider for videos using the ffmpeg tool.

## Public Functions

**ImageMagickPdfThumbnailProvider** ()

void **getThumbnail** (*sh::filesystem::Operation* \*operation,  
std::shared\_ptr<*sh::filesystem::FilesystemNode*> node, QString  
contentType, int width, int height, QIcon \*icon) **override**

### Public Static Functions

void **doInitialize** ()

void **doShutdown** ()

### Private Members

QString **pathToImagemagickConvertTool**

### Private Static Attributes

std::shared\_ptr<sh::configuration::ConfigurationValue> **cfgvalImagemagickConvertPath** = sh::configurat

**class PlaintextThumbnailProvider : public sh::tools::ThumbnailProvider**  
#include <plaintextthumbnailprovider.h> Thumbnail provider for plain text.

### Public Functions

**PlaintextThumbnailProvider** ()

void **getThumbnail** (sh::filesystem::Operation \*operation,  
std::shared\_ptr<sh::filesystem::FilesystemNode> node, QString  
contentType, int width, int height, QIcon \*icon) **override**

### Public Static Functions

void **doInitialize** ()

void **doShutdown** ()

### Private Members

QColor **brandingcolor**

### namespace ui

User interface.

This is the presentation layer. The classes directly from here are abstract; often even technically, but clearly from conceptual perspective. Find non-abstract user interface implementations in sub-namespaces.

### Typedefs

**typedef** QList<sh::ui::FilePropertyDialogTabActionsView\*> **TabViewStructWidgets**

**typedef** QPair<std::shared\_ptr<sh::ui::FilePropertyDialogTab>, TabViewStructWidgets> **TabViewStruct**

## Enums

**enum ExceptionDialogResult**

Enumeration of decisions the user can make in an *ExceptionDialog*.

*Values:*

**enumerator Shutdown**

**enumerator Close**

**enumerator Cancel**

**enumerator Retry**

**enum FileViewMode**

*Values:*

**enumerator IconView**

**enumerator ListView**

**class \_ActionExecutionInfoPanel\_HelperQObject : public QObject**

*#include <actionexecutioninfopanel.h>* A helper for some signal/slot stuff of in *ActionExecutionInfoPanel*.

## Public Functions

**void \_emit\_clicked()**

**void \_emit\_visibilityChanged()**

## Signals

**void clicked()**

**void visibilityChanged()**

**class \_FilePropertyDialogTabActionsView\_HelperQObject : public QObject**

*#include <filepropertydialogtabactionsview.h>* A helper for some signal/slot stuff of in *FilePropertyDialogTabActionsView*.

## Public Functions

**void \_emit\_buttonTriggered(int i)**

## Signals

**void buttonTriggered(int i)**

**class \_MainWindow\_HelperQObject : public QObject**

*#include <mainwindow.h>* A helper for some signal/slot stuff of in *MainWindow*.

## Public Functions

```
void _emit_currentDirectoryChanged ()
void _emit_globalViewOptionsChanged ()
void _emit_currentProfileChanged ()
void _emit_currentFileViewChanged ()
void _emit_fileViewCountChanged ()
void _emit_fileViewOptionsChanged (int i)
```

## Signals

```
void currentDirectoryChanged ()
void globalViewOptionsChanged ()
void currentProfileChanged ()
void currentFileViewChanged ()
void fileViewCountChanged ()
void fileViewOptionsChanged (int)
```

```
class _SearchPanelConfiguration_HelperQObject : public QObject
    #include <searchpanelconfiguration.h> A helper for some signal/slot stuff of in SearchPanelConfiguration.
```

```
class AboutDialog : public sh::ui::Dialog
    #include <aboutdialog.h> The 'About' dialog.

    Subclassed by sh::ui::qt::QtAboutDialog, sh::ui::web::WebAboutDialog
```

## Public Functions

```
AboutDialog ()
```

```
qint64 dialogId ()
```

Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

```
bool isInitiated ()
```

Returns if this dialog is initialized.

Must be called in main thread.

```
bool wasAccepted ()
```

Returns if this dialog was 'accepted' by the user (typically by clicking some 'OK' button).

Must be called in main thread.

```
void waitClosed ()
```

Wait until the user closed the dialog in some way.

May be called in any thread.



void **close** ()

Closes the dialog.

Must be called in main thread.

bool **wasClosed** ()

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \***manager** ()

Returns the *DialogManager* which hosts this dialog.

**class ActionExecutionInfoDialog** : public *sh::actions::ActionExecutionUserFeedback*  
*#include <actionexecutioninfodialog.h>* Progress dialog for action executions.

Subclassed by *sh::ui::qt::QtActionExecutionInfoDialog*, *sh::ui::web::WebActionExecutionInfoDialog*

## Public Types

enum **MessageBoxButton**

Buttons in a message box from *ActionExecutionUserFeedback*.

*Values:*

enumerator **NONE** = 0

enumerator **OK** = 1 << 0

enumerator **Continue** = 1 << 1

enumerator **Cancel** = 1 << 2

enumerator **Retry** = 1 << 3

enumerator **Yes** = 1 << 4

enumerator **No** = 1 << 5

## Public Functions

**ActionExecutionInfoDialog** (*sh::actions::ActionExecutionInfo* \**info*)

void **setDetails** (QString *fv*, QString *fob*, QString *tv*, QString *tob*) = 0

Sets the item details text ('from a/foo.jpg', 'to b/foo.jpg'). .

void **setHead** (QString *txt*) = 0

Sets the header text. .

void **setProgress** (bool *isDeterminate*, quint64 *done*, quint64 *all*, QString *text*) = 0

Sets the progress. .

bool **isLogicallyVisible** ()

Returns if this dialog is logically visible (i.e. set visible by the action).

void **setLogicallyVisible** (bool *v*)

Sets if this dialog is logically visible (i.e. set visible by the action). .

bool **isBackground** ()

Returns if this dialog is currently in background mode (i.e. not visible).

void **setBackground** (bool *v*)

Sets if this dialog is currently in background mode (i.e. not visible). .

```
void setForceForeground (bool v)
    Sets if this dialog is currently forced to be visible in foreground. .

int messageBox (QString text, QList<QString> answers, QString icon = QString(), int defaultanswer = -1, int cancelanswer = -1, QList<QString> answericons = QList<QString>()) = 0

int inputBox (QString text, QList<QString> answers, QString *value, QString icon = QString(), int defaultanswer = -1, int cancelanswer = -1, int valuePreselectFrom = -1, int valuePreselectTo = -1) = 0

int multilineInputBox (QString text, QList<QString> answers, QString *value, QString icon = QString(), int defaultanswer = -1, int cancelanswer = -1) = 0

int simpleChooserGridform (QString text, GridformEntries *entries, QStringList answers, int defaultanswer = -1, int cancelanswer = -1) = 0

bool credentialsDialog (QString text, bool showDomain, bool showUsername, bool showPassword, bool showAnonymous, bool showRemember, QString *domain, QString *username, QString *password, bool *isAnonymous, bool *isRemember) = 0

bool unixPermissionsDialog (bool *userMayRead, bool *userMayWrite, bool *userMayExecute, bool *groupMayRead, bool *groupMayWrite, bool *groupMayExecute, bool *othersMayRead, bool *othersMayWrite, bool *othersMayExecute, bool *sticky, bool *setuid, bool *setgid, QStringList users, QStringList groups, QString *ownerUser, QString *ownerGroup) = 0

QString simpleInputBox (QString text, QString deflt, int valuePreselectFrom = -1, int valuePreselectTo = -1)

int simpleMessageBox (QString text, int buttons = (int)MessageBoxButton::OK, QString icon = QString(), int defaultbutton = (MessageBoxButton)0, int cancelbutton = (MessageBoxButton)0)

class ActionExecutionInfoPanel
    #include <actionexecutioninfopanel.h> Abstract class for a status bar info-panel for an action execution.

    Subclassed by sh::ui::qt::QtActionExecutionInfoPanel, sh::ui::web::WebActionExecutionInfoPanel
```

## Public Functions

```
void setLabel (QString s) = 0
    Sets the label text. .

void setProgress (bool isProgressDeterminate, quint64 progressDone, quint64 progressAll) = 0
    Sets the progress. .

void setForceForeground (bool v) = 0
    Sets if the associated action is currently forced to be visible in foreground. .

void setPanelVisible (bool v) = 0
    Sets if the panel is visible. .

void setWidth (int width) = 0
    Sets the panel with (in pixels). .
```

void **onClicked** (std::function<void>  
     > *fcn*QObject \*owner = 0) Sets a handler for a click on the button (optionally bound to an owner lifetime).

void **onDestroyed** (std::function<void>  
     > *fcn*QObject \*owner = 0) Sets a handler for panel removal (optionally bound to an owner lifetime).

void **onVisibilityChanged** (std::function<void>  
     > *fcn*QObject \*owner = 0) Sets a handler for panel visibility changes (optionally bound to an owner lifetime).

**~ActionExecutionInfoPanel** ()

**class ColumnDimensions**

*#include <columndimensions.h>* Auxiliary data structure for persistent column dimensions.

It stores the width of a column whenever it changes and restores it when needed (mostly when Shallot starts).

### Public Functions

**ColumnDimensions** (int *index*)

Constructed only by the infrastructure and made available otherwise.

void **store** ()

bool **remove** ()

void **setWidth** (QString *col*, int *width*)

int **getWidth** (QString *col*, int *defaultval*)

### Public Members

int **index**

### Private Functions

std::unique\_ptr<QSettings> **db** ()

### Private Members

QHash<QString, int> **dims**

**class Dialog**: public std::enable\_shared\_from\_this<*Dialog*>

*#include <dialog.h>* Abstract subclass for Shallot dialogs (in child windows).

Subclasses of *Dialog* represent particular dialog types (like *ManageBookmarksDialog*).

For each ui mode, there typically is an implementation for all of these types, implementing one of this dialog types subclasses (mentioned before) and also some ui mode specific class.

Note: Unless stated otherwise, all methods must be called from main thread!

Note: It's not allowed to show one web dialog instance more than once. After closing it, it's sole use is to fetch answer data from it.

Note: You should not create instances directly. It's required to use *DialogManager::createAndShowDialog()* for actually showing those dialogs. Even this should not be used directly, because call depends on the current ui mode. Use the *show\*()* methods in *sh::ui::MainWindow* for creating dialogs in a ui mode independent way.

Subclassed by *sh::ui::AboutDialog*, *sh::ui::ExceptionDialog*, *sh::ui::FilePropertyDialog*, *sh::ui::LogViewDialog*, *sh::ui::ManageBookmarksDialog*, *sh::ui::ManageProfilesDialog*, *sh::ui::OpenWithDialog*, *sh::ui::StoreProfileDialog*, *sh::ui::TuningDialog*

## Public Functions

**Dialog()**

**~Dialog()**

qint64 **dialogId()**

Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

bool **isInitiated()**

Returns if this dialog is initialized.

Must be called in main thread.

bool **wasAccepted()**

Returns if this dialog was 'accepted' by the user (typically by clicking some 'OK' button).

Must be called in main thread.

void **waitClosed()**

Wait until the user closed the dialog in some way.

May be called in any thread.

void **close()**

Closes the dialog.

Must be called in main thread.

bool **wasClosed()**

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \***manager()**

Returns the *DialogManager* which hosts this dialog.

## Private Functions

void **setup**(*DialogManager* \**manager*, qint64 *dialogId*)

Sets manager and id for this dialog (right after creation).

## Private Members

*DialogManager* \***\_manager**

qint64 **\_dialogId** = -1

bool **\_closed** = false

bool **\_initied** = false

## Friends

**friend class** MainWindow

**friend class** DialogManager

## class DialogManager

*#include <dialog.h>* Creates and drives dialogs (in child windows), i.e. instances of *Dialog*.

Subclass it for implementing the low level mechanisms for handling dialogs in a particular ui mode (e.g. qt, web).

The infrastructure will use one of this implementations for creating and managing most kinds of dialogs.

Subclassed by *sh::ui::qt::QtDialogManager*, *sh::ui::web::WebDialogManager*

## Public Functions

std::shared\_ptr<*Dialog*> **getDialogById** (qint64 *id*)

Returns a *Dialog* by dialog id.

Must be called in main thread.

QList<std::shared\_ptr<*Dialog*>> **getAllDialogs** ()

Returns a list of all dialogs currently shown (in order of creation).

Must be called in main thread.

QList<qint64> **getAllDialogIds** ()

Returns a list of dialog ids of all dialogs currently shown (in order of creation).

Must be called in main thread.

template<class **TDlg**, typename ...**Args**>

std::shared\_ptr<*TDlg*> **createAndShowDialog** (*Args*... *args*)

Creates and shows a *Dialog* by class and constructor parameters.

Those dialogs (TDlg) have to implement *Dialog* (typically a subclass of it, representing a particular dialog, like *FilePropertyDialog*), and also some ui mode (e.g. qt, web) specific class (depends on that ui mode).

You typically should not have to use it outside of *MainWindow* or subclasses. The *MainWindow* interface allows to create all available dialogs in a ui mode independent way.

May be called in any thread.

**Return** The created *Dialog* instance.

**~DialogManager** ()

### Private Functions

void **closeDialog** (std::shared\_ptr<*Dialog*> *dialog*)

Closes this dialog, including internal bookkeeping.

Must be called in main thread.

void **initAndShowDialog** (std::shared\_ptr<*Dialog*> *dialog*)

Initializes and shows a freshly created dialog.

Must be called in main thread.

bool **wasAccepted** (std::shared\_ptr<*Dialog*> *dialog*) = 0

Returns if the dialog was ‘accepted’ by the user (typically by clicking some ‘OK’ button).

The result is undefined before the dialog was closed.

Must be called in main thread.

void **waitClosed** (std::shared\_ptr<*Dialog*> *dialog*) = 0

Wait until the user closed the dialog in some way.

May be called in any thread.

void **stopDialog** (std::shared\_ptr<*Dialog*> *dialog*) = 0

This method implements the ui mode specific mechanism for stopping (i.e. closing) a dialog.

Must be called in main thread.

void **showDialog** (std::shared\_ptr<*Dialog*> *dialog*) = 0

This method implements the ui mode specific mechanism for showing a dialog.

Must be called in main thread.

### Private Members

QHash<qint64, std::shared\_ptr<*Dialog*>> **\_openDialogs**

qint64 **\_nextDialogId** = 0

### Friends

**friend class** Dialog

**class** **ExceptionDialog**: public *sh::ui::Dialog*

*#include <exceptiondialog.h>* The ‘Exception’ dialog.

Subclassed by *sh::ui::qt::QtExceptionDialog*, *sh::ui::web::WebExceptionDialog*

### Public Functions

**ExceptionDialog** (QString *error1*, QString *error2*, QString *details*, QString *icon*, bool *mayRetry*, bool *mayClose*, bool *mayCancel*, bool *showLoglabelAndDetails*)

#### Parameters

- *error1*: The 1st error text.
- *error2*: The 2nd error text.
- *details*: The error details.
- *icon*: An icon (as name resolveable by *sh::base::IconManager*).
- *mayRetry*: If it’s allowed to retry.

- `mayClose`: If it's allowed to just close and continue without closing Shallot.
- `mayCancel`: If it's allowed to cancel, i.e. throwing a *`sh::exceptions::CancelException`*.
- `showLoglabelAndDetails`: If the dialog shall allow to read the details.

QString **error1** ()

Returns the 1st error text.

QString **error2** ()

Returns the 2nd error text.

QString **details** ()

Returns the error details.

QString **icon** ()

Returns the icon (as name resolveable by *`sh::base::IconManager`*).

bool **mayRetry** ()

Returns if it's allowed to retry.

bool **mayClose** ()

Returns if it's allowed to just close and continue without closing Shallot.

bool **mayCancel** ()

Returns if it's allowed to cancel, i.e. throwing a *`sh::exceptions::CancelException`*.

bool **showLoglabelAndDetails** ()

Returns if the dialog shall allow to read the details.

*ExceptionDialogResult* **answer** () = 0

Returns the answer the user has given in this dialog.

qint64 **dialogId** ()

Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

bool **isInitied** ()

Returns if this dialog is initialized.

Must be called in main thread.

bool **wasAccepted** ()

Returns if this dialog was 'accepted' by the user (typically by clicking some 'OK' button).

Must be called in main thread.

void **waitClosed** ()

Wait until the user closed the dialog in some way.

May be called in any thread.

void **close** ()

Closes the dialog.

Must be called in main thread.

bool **wasClosed** ()

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \***manager** ()

Returns the *DialogManager* which hosts this dialog.

## Private Members

```
QString _error1
QString _error2
QString _details
QString _icon
bool _mayRetry
bool _mayClose
bool _mayCancel
bool _showLoglabelAndDetails
```

```
class FilePropertyDialog: public sh::ui::Dialog
    #include <filepropertydialog.h> The ‘File Properties’ dialog.

    Subclassed by sh::ui::qt::QtFilePropertyDialog, sh::ui::web::WebFilePropertyDialog
```

## Public Functions

```

~FilePropertyDialog ()
FilePropertyDialog (QList<std::shared_ptr<sh::filesystem::FilesystemNode>> nodes)
void refresh () = 0
    Refreshes the dialog content.
QList<std::shared_ptr<FilePropertyDialogTab>> tabs ()
    Returns a list of all tabs.
sh::ui::FilePropertyDialogTabActionsView *widgetAt (std::shared_ptr<FilePropertyDialogTab>
                                                    tab, int i)
    Returns the widget from a tab at a certain index.
int widgetCount (std::shared_ptr<FilePropertyDialogTab> tab)
    Returns the number of widgets in a tab.
sh::ui::FilePropertyDialogTabTableView *createTabViewTable () = 0
    Creates a new tab view table sub-widget.
sh::ui::FilePropertyDialogTabTextView *createTabViewText () = 0
    Creates a new tab view text sub-widget.
sh::ui::FilePropertyDialogTabIconTextBannerView *createTabViewIconTextBanner ()
                                                    =
                                                    0
    Creates a new tab view icon text banner sub-widget.
qint64 dialogId ()
    Returns the dialog id.
    Each instance has an id unique in the complete Shallot process lifetime.
    Must be called in main thread.
bool isInitiated ()
    Returns if this dialog is initialized.
    Must be called in main thread.

```



bool **wasAccepted** ()  
 Returns if this dialog was ‘accepted’ by the user (typically by clicking some ‘OK’ button).  
 Must be called in main thread.

void **waitClosed** ()  
 Wait until the user closed the dialog in some way.  
 May be called in any thread.

void **close** ()  
 Closes the dialog.  
 Must be called in main thread.

bool **wasClosed** ()  
 Returns if this dialog was closed.  
 Must be called in main thread.

*DialogManager* \***manager** ()  
 Returns the *DialogManager* which hosts this dialog.

## Public Static Functions

void **addTabFactory** (int *i*, std::shared\_ptr<*FilePropertyDialogTabFactory*> *factory*)  
 Adds a *FilePropertyDialogTabFactory* so the Properties dialogs show its content.  
 See also the REGISTER\_FILEPROPERTYDIALOGTAB macro.

### Parameters

- *i*: An index which controls the display order. Use one of the REGISTER\_FILEPROPERTYDIALOGTAB\_INDEX\_\* values in *FilePropertyDialogTabFactory* as base values.

## Private Members

QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> **\_nodes**  
 QList<*TabViewStruct*> **\_tabs**

## Private Static Attributes

QHash<int, std::shared\_ptr<*FilePropertyDialogTabFactory*>> **\_propertytabs**

## Friends

**friend class** *FilePropertyDialogTab*

**class** *FilePropertyDialogTab* : **public** std::enable\_shared\_from\_this<*FilePropertyDialogTab*>  
*#include <filepropertydialog.h>* Abstract base class for one tab in the Properties dialog (containing a list of widgets).

Subclass and register this class for providing additional content in the Properties dialog.

Register it by using the REGISTER\_FILEPROPERTYDIALOGTAB macro (or by adding a *FilePropertyDialogTabFactory* via *FilePropertyDialog::addTabFactory*).

Subclassed by *sh::filepropertydialogtabs::FilePropertyDialogTabExtendedAttributes*,  
*sh::filepropertydialogtabs::FilePropertyDialogTabGeneral*, *sh::filepropertydialogtabs::FilePropertyDialogTabUnixPer*  
*sh::filepropertydialogtabs::FilePropertyDialogTabWindows*, *sh::scripting::api::ApiFilePropertyDialogTab*

## Public Functions

QString **title** () = 0

The tab title. .

QString **titleWithoutMnemonic** ()

The tab title without mnemonic.

QList<QString> **properties** () = 0

Returns the list of property labels (one entry for each widget).

Each property is displayed with a header and a widget (created in `createWidget`) with some content (populated in `updateWidget`).

void **populateWidget** (int *i*, *FilePropertyDialogTabActionsView* \**widget*) = 0

Populates the tab widget for the *i*-th property.

Typically uses the `FilePropertyDialog::create*` methods to create sub-widgets.

See also *dialog*()).

void **updateWidget** (int *i*, *FilePropertyDialogTabActionsView* \**widget*,  
*sh::filesystem::Operation* \**op*) = 0

Populates the widget for the *i*-th property with actual content. .

void **refresh** ()

Refreshes the complete content.

Typically called after some user actions in a property widget require that all the other content must be refreshed.

QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> **nodes** ()

The nodes to show.

*FilePropertyDialogTabActionsView* \***widgetAt** (int *i*)

Returns the widget for the *i*-th property.

int **widgetCount** ()

Returns the number of widgets.

std::shared\_ptr<*FilePropertyDialog*> **dialog** ()

Returns the associated dialog.

**~FilePropertyDialogTab** ()

## Private Members

std::weak\_ptr<*FilePropertyDialog*> **\_dialog**

## Friends

**friend class** FilePropertyDialog

**friend class** FilePropertyDialogTabFactoryByFunction

**friend class** sh::scripting::api::ApiFilePropertyDialogTabFactory

**class** FilePropertyDialogTabActionsView

*#include <filepropertydialogtabactionsview.h>* A tab view which shows a main widget in the main part and some buttons below.

Subclassed by *sh::ui::qt::QtFilePropertyDialogTabActionsView*, *sh::ui::web::WebFilePropertyDialog::WebFilePropertyDialogTabActionsView*

## Public Functions

**FilePropertyDialogTabActionsView** ()

Is intended to be directly constructed from everywhere.

*FilePropertyDialogTabViewContent* \***content** ()

The main widget.

void **setContent** (*FilePropertyDialogTabViewContent* \*cnt)

Sets the main widget. .

QList<QString> **buttons** ()

The list of buttons.

void **setButtons** (QList<QString> buttons)

Sets the list of buttons. .

void **setVisible** (bool v) = 0

Sets the view visible or hidden. .

void **onButtonTriggered** (std::function<void> int i

> fct, QObject \*owner = 0) Sets a handler for a click on one of the buttons (optionally bound to an owner lifetime).

**class** FilePropertyDialogTabFactory

*#include <filepropertydialog.h>* Abstract factory for creating property dialog tabs for the selected nodes.

Register an instance of this class with *FilePropertyDialog::addTabFactory* in order to make a property tab implementation available. See also *FilePropertyDialogTabFactoryByFunction*.

Subclassed by *sh::scripting::api::ApiFilePropertyDialogTabFactory*, *sh::ui::FilePropertyDialogTabFactoryByFunction*

## Public Functions

std::shared\_ptr<*sh::ui::FilePropertyDialogTab*> **construct** (std::shared\_ptr<*sh::ui::FilePropertyDialog*> dialog) = 0

Constructs a fresh instance of a *FilePropertyDialogTab* implementation. .

**~FilePropertyDialogTabFactory** ()

### Public Static Attributes

**const** int **REGISTER\_FILEPROPERTYDIALOGTAB\_INDEX\_CORE** = 10000

Base value for display indexes of core (i.e. maximum important) tabs.

Used in *FilePropertyDialog::addTabFactory*.

**const** int **REGISTER\_FILEPROPERTYDIALOGTAB\_INDEX\_VERYIMPORTANT** = 20000

Base value for display indexes of very important tabs.

Used in *FilePropertyDialog::addTabFactory*.

**const** int **REGISTER\_FILEPROPERTYDIALOGTAB\_INDEX\_IMPORTANT** = 30000

Base value for display indexes of important tabs.

Used in *FilePropertyDialog::addTabFactory*.

**const** int **REGISTER\_FILEPROPERTYDIALOGTAB\_INDEX\_NORMAL** = 40000

Base value for display indexes of tabs with medium importance.

Used in *FilePropertyDialog::addTabFactory*.

**const** int **REGISTER\_FILEPROPERTYDIALOGTAB\_INDEX\_EXOTIC** = 50000

Base value for display indexes of exotic (i.e. less important) tabs.

Used in *FilePropertyDialog::addTabFactory*.

**class** **FilePropertyDialogTabFactoryByFunction** : **public** *sh::ui::FilePropertyDialogTabFactory*  
#include <filepropertydialog.h> Implementation of *FilePropertyDialogTabFactory* for making custom *FilePropertyDialogTab* implementations available.

This implementation relies on an external factory function which must be provided to the constructor.

Used internally inside the REGISTER\_FILEPROPERTYDIALOGTAB macro.

### Public Functions

**FilePropertyDialogTabFactoryByFunction** (std::function<std::shared\_ptr<*sh::ui::FilePropertyDialogTab*>  
> *fcn*)

std::shared\_ptr<*sh::ui::FilePropertyDialogTab*> **construct** (std::shared\_ptr<*sh::ui::FilePropertyDialog*>  
*dialog*)

Constructs a fresh instance of a *FilePropertyDialogTab* implementation. .

### Public Static Attributes

**const** int **REGISTER\_FILEPROPERTYDIALOGTAB\_INDEX\_CORE** = 10000

Base value for display indexes of core (i.e. maximum important) tabs.

Used in *FilePropertyDialog::addTabFactory*.

**const** int **REGISTER\_FILEPROPERTYDIALOGTAB\_INDEX\_VERYIMPORTANT** = 20000

Base value for display indexes of very important tabs.

Used in *FilePropertyDialog::addTabFactory*.

**const** int **REGISTER\_FILEPROPERTYDIALOGTAB\_INDEX\_IMPORTANT** = 30000

Base value for display indexes of important tabs.

Used in *FilePropertyDialog::addTabFactory*.

**const** int **REGISTER\_FILEPROPERTYDIALOGTAB\_INDEX\_NORMAL** = 40000

Base value for display indexes of tabs with medium importance.

Used in *FilePropertyDialog::addTabFactory*.

**const** int **REGISTER\_FILEPROPERTYDIALOGTAB\_INDEX\_EXOTIC** = 50000

Base value for display indexes of exotic (i.e. less important) tabs.

Used in *FilePropertyDialog::addTabFactory*.

## Private Members

**std::function< std::shared\_ptr< sh::ui::FilePropertyDialogTab >> fct**

**class FilePropertyDialogTabIconTextBannerView**: public *sh::ui::FilePropertyDialogTabViewContent*  
*#include <filepropertydialogtabactionsview.h>* A tab view for showing texts and icons combined.

Subclassed by *sh::ui::qt::QtFilePropertyDialogTabIconTextBannerView*,  
*sh::ui::web::WebFilePropertyDialog::WebFilePropertyDialogTabIconTextBannerView*

## Public Functions

**FilePropertyDialogTabIconTextBannerView()**

Is intended to be directly constructed from everywhere.

void **clear** () = 0

Clears the contents.

void **insertIcon** (QIcon *icon*, int *sizePt* = 24) = 0

Adds an icon.

void **insertText** (QString *text*) = 0

Adds a text.

**class FilePropertyDialogTabTableView**: public *sh::ui::FilePropertyDialogTabViewContent*  
*#include <filepropertydialogtabactionsview.h>* A tab view for showing key/value pairs.

Subclassed by *sh::ui::qt::QtFilePropertyDialogTabTableView*, *sh::ui::web::WebFilePropertyDialog::WebFilePropertyDialogTabTableView*

## Public Types

**typedef** QPair<int, int> **ItemIndex**

## Public Functions

**FilePropertyDialogTabTableView()**

Is intended to be directly constructed from everywhere.

void **setContent** (QList<QPair<QString, QString>> *content*) = 0

Sets the content.

QList<*ItemIndex*> **getSelectedItems** () = 0

Returns the selected cells.

QVariant **getItemValue** (int *r*, int *c*) = 0

Returns a value of a cell.

```
class FilePropertyDialogTabTextView : public sh::ui::FilePropertyDialogTabViewContent
    #include <filepropertydialogtabactionsview.h> A tab view for showing a text.
```

Subclassed by *sh::ui::qt::QtFilePropertyDialogTabTextView*, *sh::ui::web::WebFilePropertyDialog::WebFilePropertyD*

### Public Functions

```
FilePropertyDialogTabTextView ()
```

Is intended to be directly constructed from everywhere.

```
void setContent (QString content) = 0
```

Sets the content.

```
class FilePropertyDialogTabViewContent
```

*#include <filepropertydialogtabactionsview.h>* Abstract subclass for a tab view content.

Subclassed by *sh::ui::FilePropertyDialogTabIconTextBannerView*,  
*sh::ui::FilePropertyDialogTabTableView*, *sh::ui::FilePropertyDialogTabTextView*

### Public Functions

```
FilePropertyDialogTabViewContent ()
```

```
~FilePropertyDialogTabViewContent ()
```

```
class FileView : public QObject
```

*#include <fileview.h>* Abstract base class for a file view implementation (i.e. which shows the content of a directory somehow).

Subclassed by *sh::ui::qt::QtFileViewControl*, *sh::ui::web::WebFileView*

### Public Functions

```
FileView ()
```

```
~FileView ()
```

```
sh::ui::ColumnDimensions *columnDimensions () = 0
```

Returns the column dimensions handler. .

```
sh::tools::HistoryTracker<std::shared_ptr<const sh::filesystem::Eurl>> *historyTracker ()
                                                                    =
                                                                    0
```

Returns the history tracker. .

```
FileViewMode viewmode () = 0
```

Returns the view mode (icons, list, ... ?). .

```
void setViewmode (FileViewMode m) = 0
```

Sets the view mode. .

```
int index () = 0
```

Returns the position of this file view within the panel. .

```
std::shared_ptr<sh::filesystem::FilesystemNode> node ()
```

Returns the current directory.

```

void gotoDir (std::shared_ptr<sh::filesystem::FilesystemNode> node)
    Jumps to a new current directory.

    Implementations have to call the base class implementation inside.

sh::filesystem::SizeFormatting getSizeFormattingMode () = 0
    Returns the mode how file sizes are formatted for displaying. .

void setSizeFormattingMode (sh::filesystem::SizeFormatting mode) = 0
    Sets the mode how file sizes are formatted for displaying. .

bool hiddenFilesVisible () = 0
    Returns if hidden files are visible. .

void setHiddenFilesVisible (bool v) = 0
    Sets the visibility of hidden files. .

int iconDimension () = 0
    Returns the icon size (in pixels). .

void setIconDimension (int v) = 0
    Sets the icon size (in pixels). .

void setSort (int column, Qt::SortOrder order) = 0
    Sets how to sort this view. .

int sortColumn () = 0
    Returns the index of the current sort column. .

Qt::SortOrder sortOrder () = 0
    Returns the current sort order. .

QList<std::shared_ptr<sh::filesystem::FilesystemNode>> selectedNodes () = 0
    Returns the nodes which the user has selected in this fileview. .

void setSelection (const QList<std::shared_ptr<sh::filesystem::FilesystemNode>>
                  nodes) = 0
    Sets the node selection of this fileview. .

QList<std::shared_ptr<sh::filesystem::FilesystemNode>> getAllVisibleNodes () = 0
    Returns a list of all nodes currently listed in this file view. .

void reload (bool skipModel = false)
    Reloads the data inside this file view.

sh::filesystem::FilesystemModelFileviewProxy *filemodel ()
    Returns the filesystem model for this view. .

void setFilemodel (sh::filesystem::FilesystemModelFileviewProxy *model)
    Sets the filesystem model for this view. .

```

## Signals

```

void selectionChanged ()
    Triggered when the selection have changed.

void viewOptionChanged ()
    Triggered when some view options have changed.

```

## Private Members

*sh::filesystem::FilesystemModelFileviewProxy* \*\_**model** = nullptr

std::shared\_ptr<QObject> **\_visibleviewslifetimecanary** = nullptr

std::shared\_ptr<*sh::filesystem::FilesystemNode*> **\_node** = nullptr

**class LogViewDialog : public *sh::ui::Dialog***

*#include <logviewdialog.h>* The ‘Log’ dialog..

Subclassed by *sh::ui::qt::QtLogViewDialog*, *sh::ui::web::WebLogViewDialog*

## Public Functions

**LogViewDialog** (QString *headtext*, QString *subheadtxt*, *sh::base::LogSeverity* *minseverity*)

### Parameters

- *headtext*: The header text.
- *subheadtxt*: The 2nd header text.
- *minseverity*: The minimum message severity this dialog shows initially (the user can typically change that later on).

QString **headtext** ()

Returns the header text.

QString **subheadtxt** ()

Returns the 2nd header text.

*sh::base::LogSeverity* **minimumSeverity** ()

Returns the minimum severity.

qint64 **dialogId** ()

Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

bool **isInited** ()

Returns if this dialog is initialized.

Must be called in main thread.

bool **wasAccepted** ()

Returns if this dialog was ‘accepted’ by the user (typically by clicking some ‘OK’ button).

Must be called in main thread.

void **waitClosed** ()

Wait until the user closed the dialog in some way.

May be called in any thread.

void **close** ()

Closes the dialog.

Must be called in main thread.

bool **wasClosed** ()

Returns if this dialog was closed.

Must be called in main thread.



*DialogManager* \*manager ( )

Returns the *DialogManager* which hosts this dialog.

## Private Members

QString \_headtext

QString \_subheadtxt

*sh::base::LogSeverity* \_minseverity

## class MainWindow

#include <mainwindow.h> The abstract main window class. There is one instance of it, and it is also used for other ui parts (e.g. creating some dialogs, ...).

Subclassed by *sh::ui::qt::QtMainWindow*, *sh::ui::web::WebMainWindow*

## Public Functions

**MainWindow** ( )

**~MainWindow** ( )

void **\_initialize** (*sh::base::SingletonInitializer* \*singletonInitializer)

Initializes the main window. For custom initialization logic, see MainWindow.initialize. .

int **fileViewCount** ( ) = 0

Returns the current number of file views. .

void **addFileView** (bool *reinitialize* = false) = 0

Adds a new file view. .

void **removeFileView** (int *i*) = 0

Removes a file view. .

*sh::ui::FileView* \***currentFileView** ( ) = 0

Returns the file view which is currently active (i.e. focussed). .

*sh::ui::FileView* \***getFileView** (int *i*) = 0

Returns a file view by position index. .

void **fileViewsReloadAll** ( )

Reload all content in each file view.

void **onFileViewOptionsChanged** (std::function<void> int

> *fct*, QObject \**owner* = 0) Sets a handler for some view options in a file view changed (optionally bound to an owner lifetime).

void **onCurrentFileViewChanged** (std::function<void>

> *fct*, QObject \**owner* = 0) Sets a handler for the currently active file view changed (optionally bound to an owner lifetime).

void **onFileViewCountChanged** (std::function<void>

> *fct*, QObject \**owner* = 0) Sets a handler for the number of file views changed (optionally bound to an owner lifetime).

void **onCurrentDirectoryChanged** (std::function<void>

> *fct*, QObject \**owner* = 0) Sets a handler for the current directory in the active file view changed (optionally bound to an owner lifetime).

```
void onGlobalViewOptionsChanged (std::function<void>
    > fcnQObject *owner = 0)Sets a handler for some global view options changed (optionally bound
    to an owner lifetime).

void onCurrentProfileChanged (std::function<void>
    > fcnQObject *owner = 0)Sets a handler for the current profile changed (optionally bound to an
    owner lifetime).

void jumpToEurl (std::shared_ptr<const sh::filesystem::Eurl> eurl)
    Let the current view jump to another location. .

void jumpToNode (std::shared_ptr<sh::filesystem::FilesystemNode> node) = 0
    Let the current view jump to another location. .

void setTitlePattern (QString value)
    Sets the window title pattern. .

QString titlePattern ()
    Returns the window title pattern.

void setCurrentProfile (QString profile)
    Sets the current profile. .

QString currentProfile ()
    Returns the current profile.

void reloadProfile ()
    Reloads the profile data.

void setTreeVisible (bool v)
    Sets the visibility of the directory tree. .

bool treeVisible ()
    Returns the visibility of the directory tree.

void setTreeSticky (bool v)
    Sets if the directory tree should follow the active file view. .

bool treeSticky ()
    Returns if the directory tree follows the active file view.

void setFileDetailsPanelVisible (bool v)
    Sets the visibility of the file details panel. .

bool fileDetailsPanelVisible ()
    Returns the visibility of the file details panel.

std::shared_ptr<sh::filesystem::FilesystemNode> currentDirectory () = 0
    Gets the sh::filesystem::FilesystemNode selected in the current view. .

std::shared_ptr<sh::ui::ActionExecutionInfoPanel> addInfoPanel (int position,
    sh::actions::ActionExecutionInfo
    *info, QColor color =
    QColor()) = 0

    Creates a new action execution panel.

    Let this smart pointer die for removing it.

std::shared_ptr<sh::ui::ActionExecutionInfoPanel> addInfoPanel (sh::actions::ActionExecutionInfo
    *info, QColor color =
    QColor())

    Creates a new action execution panel.

    Let this smart pointer die for removing it.
```

```

void jumpbarSetTextMode () = 0
    Sets the jumpbar to text input mode. .

void jumpbarSetButtonMode () = 0
    Sets the jumpbar to button mode. .

bool jumpbarIsTextMode () = 0
    Returns if the jumpbar is in text input mode. .

QString toolbarPosition () = 0
    Returns the current toolbar position. .

void setToolBarPosition (QString pos) = 0
    Sets the toolbar position. .

QString detailPanelPosition () = 0
    Returns the current detail panel position. .

void setDetailPanelPosition (QString pos) = 0
    Sets the detail panel position. .

std::shared_ptr<AboutDialog> showAboutDialog () = 0
    Shows and returns an 'About' dialog. .

std::shared_ptr<ManageProfilesDialog> showManageProfilesDialog () = 0
    Shows and returns a 'Manage Saved Settings' dialog. .

std::shared_ptr<OpenWithDialog> showOpenWithDialog (std::shared_ptr<sh::filesystem::FileSystemNode>
                                                    node,
                                                    QList<std::shared_ptr<sh::tools::filetypes::OpenMethod>
                                                    openMethods) = 0
    Shows and returns a 'Open With' dialog. .

std::shared_ptr<StoreProfileDialog> showStoreProfileDialog (std::shared_ptr<const
                                                            sh::filesystem::Eurl>
                                                            eurl) = 0
    Shows and returns a 'Save Settings' dialog. .

std::shared_ptr<TuningDialog> showTuningDialog () = 0
    Shows and returns a 'Tuning' dialog. .

std::shared_ptr<LogViewDialog> showLogViewDialog (QString headtext = QString(),
                                                    QString subheadtxt = QString(),
                                                    sh::base::LogSeverity minseverity =
                                                    sh::base::LogSeverity::_DEFAULT)
                                                    = 0
    Shows and returns a 'Log' dialog. .

std::shared_ptr<ManageBookmarksDialog> showManageBookmarksDialog () = 0
    Shows and returns a 'Manage Bookmarks' dialog. .

std::shared_ptr<FilePropertyDialog> showFilePropertyDialog (QList<std::shared_ptr<sh::filesystem::FilesystemNode>
                                                            nodes) = 0
    Shows and returns a 'File Properties' dialog. .

std::shared_ptr<ActionExecutionInfoDialog> createActionExecutionInfoDialog (sh::actions::ActionExecutionInfo
                                                                            *info)
                                                                            =
                                                                            0
    Creates an action execution dialog. .

std::shared_ptr<ActionExecutionInfoPanel> showErrorPanel ()
    Shows an error panel.

```

`std::shared_ptr<DialogManager> dialogManager () = 0`  
Returns the *DialogManager* which creates and drives *Dialogs* for this main window.  
You typically should not need to use it directly.

`void _closeDirectly ()`  
Directly begin application shutdown without checks.

`bool closeApp ()`  
Check if the application may shut down now, and if so, initiate it.

`bool isCloseable (QString *msg = nullptr)`  
Check if the application may shut down now.

`void handleCloseAppRejected (QString rejectmsg) = 0`  
React on a rejected application close request (with some feedback message). .

`void handleClose ()`  
Do some cleanup steps just when closing starts. Implementations must call base implementation!

`bool isClosing ()`  
Returns if the application is currently closing.

## Public Static Functions

*MainWindow* \*`mainWindow ()`  
Returns the instance of this singleton.

`void setMainWindow (MainWindow *mainWindow)`  
Sets the main window instance. .

`bool isReady ()`  
Returns if this process ui is ready (i.e. is created or runs headless).

`bool runsHeadless ()`  
Returns if this process runs headless, i.e. without any ui, just for a background process.

`void _closeDirectly (sh::base::SingletonInitializer *singletonInitializer)`

`QString uiMode ()`  
Returns the UI mode (e.g. qt, web).

## Private Functions

`void _refreshIsCloseable ()`

## Private Members

`bool _closing = false`

`bool _treeSticky = true`

`bool _treeVisible = true`

`bool _detailsPanelVisible = true`

`QString _titlePattern`

`QList<std::shared_ptr<sh::ui::ActionExecutionInfoPanel>> _errorpanels`

`QString _currentProfile`

```

QString _isCloseable
bool _thumbnailrequestongoing = false
bool _thumbnailrequestfollowup = false
int _thumbnailrequestfollowup_width = 0
int _thumbnailrequestfollowup_height = 0
std::function<void ()> _thumbnailrequestfollowup_onBeforeRequest = nullptr
std::function<void (QIcon)> _thumbnailrequestfollowup_onArrived = nullptr
int _thumbnaillastreqwidth = 0
int _thumbnaillastreqheight = 0
quint64 _thumbnaillastrequestid = 0

```

### Private Static Attributes

```

MainWindow *_mainWindow = nullptr
bool _mainWindow_runheadless = false
KillHelperThread *_killthread = nullptr

```

### Friends

```

friend class sh::exceptions::Exception
class KillHelperThread : public QThread
    Helps application shutdown in some situations :).

```

### Public Functions

```

void run ()

class ManageBookmarksDialog : public sh::ui::Dialog
    #include <managebookmarksdialog.h> The 'Manage Bookmarks' dialog.
    Subclassed by sh::ui::qt::QtManageBookmarksDialog, sh::ui::web::WebManageBookmarksDialog

```

### Public Functions

```

ManageBookmarksDialog ()

quint64 dialogId ()
    Returns the dialog id.
    Each instance has an id unique in the complete Shallot process lifetime.
    Must be called in main thread.

bool isInitiated ()
    Returns if this dialog is initialized.
    Must be called in main thread.

```

bool **wasAccepted** ()

Returns if this dialog was ‘accepted’ by the user (typically by clicking some ‘OK’ button).

Must be called in main thread.

void **waitClosed** ()

Wait until the user closed the dialog in some way.

May be called in any thread.

void **close** ()

Closes the dialog.

Must be called in main thread.

bool **wasClosed** ()

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \***manager** ()

Returns the *DialogManager* which hosts this dialog.

**class ManageProfilesDialog**: public *sh::ui::Dialog*

*#include <manageprofilesdialog.h>* The ‘Manage Saved Settings’ dialog.

Subclassed by *sh::ui::qt::QtManageProfilesDialog*, *sh::ui::web::WebManageProfilesDialog*

## Public Functions

**ManageProfilesDialog** ()

qint64 **dialogId** ()

Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

bool **isInitiated** ()

Returns if this dialog is initialized.

Must be called in main thread.

bool **wasAccepted** ()

Returns if this dialog was ‘accepted’ by the user (typically by clicking some ‘OK’ button).

Must be called in main thread.

void **waitClosed** ()

Wait until the user closed the dialog in some way.

May be called in any thread.

void **close** ()

Closes the dialog.

Must be called in main thread.

bool **wasClosed** ()

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \***manager** ( )

Returns the *DialogManager* which hosts this dialog.

**class** **OpenWithDialog**: **public** *sh::ui::Dialog*

#include <openwithdialog.h> The ‘Open With’ dialog.

Subclassed by *sh::ui::qt::QtOpenWithDialog*, *sh::ui::web::WebOpenWithDialog*

## Public Functions

**OpenWithDialog** (std::shared\_ptr<*sh::filesystem::FileSystemNode*> *node*,  
QList<std::shared\_ptr<*sh::tools::filetypes::OpenMethod*>> *openMethods*)

### Parameters

- *node*: The file which is later to be opened.
- *openMethods*: The open-methods (programs for opening the file) to present for choice.

std::shared\_ptr<*sh::tools::filetypes::OpenMethod*> **chosenMethod** ( ) = 0

Returns the chosen open-method (program for opening the file).

bool **rememberForMimetype** ( ) = 0

Returns if the chosen method is checked to be remembered for the same mime-type in the future.

std::shared\_ptr<const *sh::filesystem::Eurl*> **rememberForDirectory** ( ) = 0

If it's checked for the chosen method to be remembered for some subdirectory in the future, it returns the Eurl of this subdirectory, otherwise nullptr.

bool **rememberForFile** ( ) = 0

Returns if the chosen method is checked to be remembered for the same file in the future.

std::shared\_ptr<*sh::filesystem::FileSystemNode*> **node** ( )

Returns the file node which is later to be opened.

QList<std::shared\_ptr<*tools::filetypes::OpenMethod*>> **openMethods** ( )

Returns the open-methods (programs for opening the file) to present for choice.

qint64 **dialogId** ( )

Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

bool **isInitiated** ( )

Returns if this dialog is initialized.

Must be called in main thread.

bool **wasAccepted** ( )

Returns if this dialog was ‘accepted’ by the user (typically by clicking some ‘OK’ button).

Must be called in main thread.

void **waitClosed** ( )

Wait until the user closed the dialog in some way.

May be called in any thread.

void **close** ( )

Closes the dialog.

Must be called in main thread.

bool **wasClosed** ()  
Returns if this dialog was closed.  
Must be called in main thread.

*DialogManager* \***manager** ()  
Returns the *DialogManager* which hosts this dialog.

### Private Members

std::shared\_ptr<*sh::filesystem::FilesystemNode*> **\_node**  
QList<std::shared\_ptr<*sh::tools::filetypes::OpenMethod*>> **\_openMethods**

**class SearchPanelAbstractEditor**  
*#include <searchpanelconfiguration.h>* Abstract base class for editor widgets in a search panel.  
Subclassed by *sh::ui::SearchPanelDateTimeEditor*, *sh::ui::SearchPanelLabelEditor*,  
*sh::ui::SearchPanelSpacerEditor*, *sh::ui::SearchPanelTextEditor*

### Public Functions

**SearchPanelAbstractEditor** () = default  
**~SearchPanelAbstractEditor** () = default  
bool **isWidgetEnabled** () = 0  
Returns if the widget is enabled. .  
void **setWidgetEnabled** (bool v) = 0  
Sets if the widget is enabled. .

**class SearchPanelButton**  
*#include <searchpanelconfiguration.h>* A button in the button bar of a search panel.  
Subclassed by *sh::ui::qt::QtSearchPanelButton*

### Public Functions

**SearchPanelButton** ()  
**~SearchPanelButton** ()  
void **setButtonText** (QString txt) = 0  
Sets the button text. .  
void **setMenuSelection** (int i) = 0  
Sets the currently selected menu item (for buttons with menus). .

**class SearchPanelConfiguration**  
*#include <searchpanelconfiguration.h>* Configuration for a search panel population.  
It is populated with ui controls by the chosen search criterion. Later on those controls are used for updating the configuration data.  
Subclassed by *sh::ui::qt::QtSearchPanelConfiguration*



## Public Functions

**SearchPanelConfiguration()**

**~SearchPanelConfiguration()**

*SearchPanelButton* \***addMenuButton** (QString *text*, QStringList *menu*, std::function<void> *int* > *onchanged* = 0) Adds and returns a button for a menu. .

*SearchPanelButton* \***addActionButton** (QString *text*, std::function<void> > *action* = 0) Adds and returns a button for an action. .

*SearchPanelTextEditor* \***addTextEditor** () = 0  
Adds and returns a text editor. .

*SearchPanelDateTimeEditor* \***addDateTimeEditor** () = 0  
Adds and returns a date/time editor. .

*SearchPanelLabelEditor* \***addLabel** () = 0  
Adds and returns a label. .

*SearchPanelSpacerEditor* \***addSpacer** () = 0  
Adds and returns a spacer. .

*SearchPanelAbstractEditor* \***getEditorAt** (int *i*) = 0  
Returns the editor widget at a given position. .

void **onDestroyed** (std::function<void> > *fcn*) = 0  
Sets a handler for panel removal (optionally bound to an owner lifetime).

## Public Members

*\_SearchPanelConfiguration\_HelperQObject* **myqobject**

**class SearchPanelDateTimeEditor** : public *sh::ui::SearchPanelAbstractEditor*  
#include <searchpanelconfiguration.h> A date/time picker for search panel usage.  
Subclassed by *sh::ui::qt::QtSearchPanelAbstractEditor* < *QDateTimeEdit*, *SearchPanelDateTimeEditor* >

## Public Functions

**SearchPanelDateTimeEditor** () = default

QDateTime **datetime** () = 0  
Returns the selected date/time. .

void **setDatetime** (QDateTime *s*) = 0  
Sets the selected date/time.

bool **isWidgetEnabled** () = 0  
Returns if the widget is enabled. .

void **setWidgetEnabled** (bool *v*) = 0  
Sets if the widget is enabled. .

**class SearchPanelLabelEditor** : public *sh::ui::SearchPanelAbstractEditor*  
#include <searchpanelconfiguration.h> A text label for search panel usage.  
Subclassed by *sh::ui::qt::QtSearchPanelAbstractEditor* < *QLabel*, *sh::ui::SearchPanelLabelEditor* >

### Public Functions

**SearchPanelLabelEditor** () = default

QString **textContent** () = 0  
Returns the text content. .

void **setTextContent** (QString *s*) = 0  
Sets the text content. .

*SearchPanelAbstractEditor* \***focusProxyEditor** () = 0  
Returns the focus proxy widget (which focus gets redirected to in some situations). .

void **setFocusProxyEditor** (*SearchPanelAbstractEditor* \**w*) = 0  
Sets the focus proxy widget (which focus gets redirected to in some situations). .

bool **isWidgetEnabled** () = 0  
Returns if the widget is enabled. .

void **setWidgetEnabled** (bool *v*) = 0  
Sets if the widget is enabled. .

```
class SearchPanelSpacerEditor : public sh::ui::SearchPanelAbstractEditor
#include <searchpanelconfiguration.h> A spacer for search panel usage.

Subclassed by sh::ui::qt::QtSearchPanelAbstractEditor< QLabel, sh::ui::SearchPanelSpacerEditor
>
```

### Public Functions

**SearchPanelSpacerEditor** () = default

bool **isWidgetEnabled** () = 0  
Returns if the widget is enabled. .

void **setWidgetEnabled** (bool *v*) = 0  
Sets if the widget is enabled. .

```
class SearchPanelTextEditor : public sh::ui::SearchPanelAbstractEditor
#include <searchpanelconfiguration.h> A single line text editor for search panel usage.

Subclassed by sh::ui::qt::QtSearchPanelAbstractEditor< QLineEdit, sh::ui::SearchPanelTextEditor
>
```

### Public Functions

**SearchPanelTextEditor** () = default

QString **textContent** () = 0  
Returns the text content. .

void **setTextContent** (QString *s*) = 0  
Sets the text content. .

QString **placeholderDescription** () = 0  
Returns the placeholder description text (visible if field is empty). .

void **setPlaceholderDescription** (QString *s*) = 0  
Sets the placeholder description text (visible if field is empty). .

bool **isEnabled** () = 0  
Returns if the widget is enabled. .

void **setEnabled** (bool v) = 0  
Sets if the widget is enabled. .

**class StoreProfileDialog** : public *sh::ui::Dialog*  
#include <storeprofiledialog.h> The ‘Save Settings’ dialog.  
Subclassed by *sh::ui::qt::QtStoreProfileDialog*, *sh::ui::web::WebStoreProfileDialog*

## Public Functions

**StoreProfileDialog** (std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)

### Parameters

- eurl: The current directory this dialog shall work on.

std::shared\_ptr<const *sh::filesystem::Eurl*> **currentDirectory** ()  
Returns the current directory this dialog shall work on.

QList<*SettingEntry*> **checkedSettings** () = 0  
Returns the list of settings the user has checked to store.

QString **profileName** () = 0  
Returns the profile the user has chosen to store settings for.

bool **withSubDirectories** () = 0  
Returns if the user has chosen to apply the checked settings also for subdirectories.

QStringList **inheritsFrom** () = 0  
Returns the list of profiles the user has chosen to inherit from.

bool **hasGlobal** () = 0  
Returns if the user has chosen to apply the checked settings for each directory (instead of the current directory).

qint64 **dialogId** ()  
Returns the dialog id.  
  
Each instance has an id unique in the complete Shallot process lifetime.  
  
Must be called in main thread.

bool **isInit** ()  
Returns if this dialog is initialized.  
  
Must be called in main thread.

bool **wasAccepted** ()  
Returns if this dialog was ‘accepted’ by the user (typically by clicking some ‘OK’ button).  
  
Must be called in main thread.

void **waitClosed** ()  
Wait until the user closed the dialog in some way.  
  
May be called in any thread.

void **close** ()  
Closes the dialog.  
  
Must be called in main thread.

bool **wasClosed** ()  
Returns if this dialog was closed.  
Must be called in main thread.

*DialogManager* \***manager** ()  
Returns the *DialogManager* which hosts this dialog.

### Private Members

std::shared\_ptr<const *sh::filesystem::Eurl*> **\_eurl**

**class SettingEntry**  
*#include <storeprofiledialog.h>* A storable entry in a ‘Save Settings’ dialog.

### Public Functions

**SettingEntry** (*sh::settings::Setting* \**setting*, int *onlyInFileview* = -1)

*sh::settings::Setting* \***setting** ()  
Returns the setting to store.

int **fileview** ()  
Returns the index of the file view to store this setting for (or -1 for all).

bool **isForFileview** ()  
Returns if this setting is to be stored for a particular file view (instead of for all).

### Private Members

*sh::settings::Setting* \***\_setting**

int **\_fileview**

**class TuningDialog**: public *sh::ui::Dialog*  
*#include <tuningdialog.h>* The ‘Tuning’ dialog.  
Subclassed by *sh::ui::qt::QtTuningDialog*, *sh::ui::web::WebTuningDialog*

### Public Functions

**TuningDialog** ()

qint64 **dialogId** ()  
Returns the dialog id.  
Each instance has an id unique in the complete Shallot process lifetime.  
Must be called in main thread.

bool **isInited** ()  
Returns if this dialog is initialized.  
Must be called in main thread.

bool **wasAccepted** ()  
Returns if this dialog was ‘accepted’ by the user (typically by clicking some ‘OK’ button).  
Must be called in main thread.

void **waitClosed** ()  
 Wait until the user closed the dialog in some way.  
 May be called in any thread.

void **close** ()  
 Closes the dialog.  
 Must be called in main thread.

bool **wasClosed** ()  
 Returns if this dialog was closed.  
 Must be called in main thread.

*DialogManager* \***manager** ()  
 Returns the *DialogManager* which hosts this dialog.

**namespace qt**  
 The Qt user interface implementation.  
 This is the default UI. It builds a desktop application based on the Qt gui components.

**class LineEditWithKeyboardShortcuts** : public QLineEdit  
*#include <qtjumpbar.h>* Helper widget of *QtJumpBar*.

### Public Functions

**LineEditWithKeyboardShortcuts** (QWidget \*parent = 0)

### Signals

void **enterPressed** ()  
 void **escapePressed** ()

**class QtAboutDialog** : public *sh::ui::qt::QtDialog*, public *sh::ui::AboutDialog*  
*#include <qtaboutdialog.h>* Qt based about dialog.

### Public Functions

**QtAboutDialog** ()

**~QtAboutDialog** ()

qint64 **dialogId** ()  
 Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.  
 Must be called in main thread.

bool **isInited** ()  
 Returns if this dialog is initialized.  
 Must be called in main thread.

bool **wasAccepted** ()  
 Returns if this dialog was ‘accepted’ by the user (typically by clicking some ‘OK’ button).  
 Must be called in main thread.

void **waitClosed** ()  
Wait until the user closed the dialog in some way.  
May be called in any thread.

void **close** ()  
Closes the dialog.  
Must be called in main thread.

bool **wasClosed** ()  
Returns if this dialog was closed.  
Must be called in main thread.

*DialogManager* \***manager** ()  
Returns the *DialogManager* which hosts this dialog.

### Private Members

*Ui::QtAboutDialog* \***ui**

### Private Slots

void **on\_btnClose\_clicked** ()  
void **on\_btnLicense\_clicked** ()  
void **on\_btnHomepage\_clicked** ()

**class QtActionExecutionInfoDialog** : public QDialog, public *sh::ui::ActionExecutionInfoDialog*  
*#include <qactionexecutioninfodialog.h>* Qt progress dialog for action executions.

### Public Types

**enum MessageBoxButton**  
Buttons in a message box from *ActionExecutionUserFeedback*.  
*Values:*

**enumerator NONE** = 0  
**enumerator OK** = 1 << 0  
**enumerator Continue** = 1 << 1  
**enumerator Cancel** = 1 << 2  
**enumerator Retry** = 1 << 3  
**enumerator Yes** = 1 << 4  
**enumerator No** = 1 << 5

## Public Functions

```

QtActionExecutionInfoDialog (sh::actions::ActionExecutionInfo *info, QWidget
                               *parent = 0)

~QtActionExecutionInfoDialog ()

void setDetails (QString fv, QString fob, QString tv, QString tob)
    Sets the item details text ('from a/foo.jpg', 'to b/foo.jpg'). .

void setHead (QString txt)
    Sets the header text. .

void setProgress (bool isDeterminate, quint64 done, quint64 all, QString text)
    Sets the progress. .

int messageBox (QString text, QList<QString> answers, QString icon = QString(), int de-
                faultanswer = -1, int cancelanswer = -1, QList<QString> answericons =
                QList<QString>())

int inputBox (QString text, QList<QString> answers, QString *value, QString icon =
              QString(), int defaultanswer = -1, int cancelanswer = -1, int valuePreselect-
              From = -1, int valuePreselectTo = -1)

int multilineInputBox (QString text, QList<QString> answers, QString *value, QString
                       icon = QString(), int defaultanswer = -1, int cancelanswer = -1)

int simpleChooserGridform (QString text, GridformEntries *entries, QStringList an-
                          swers, int defaultanswer = -1, int cancelanswer = -1)

bool credentialsDialog (QString text, bool showDomain, bool showUsername, bool
                        showPassword, bool showAnonymous, bool showRemember,
                        QString *domain, QString *username, QString *password,
                        bool *isAnonymous, bool *isRemember)

bool unixPermissionsDialog (bool *userMayRead, bool *userMayWrite, bool *user-
                            MayExecute, bool *groupMayRead, bool *groupMay-
                            Write, bool *groupMayExecute, bool *othersMayRead,
                            bool *othersMayWrite, bool *othersMayExecute, bool
                            *sticky, bool *setuid, bool *setgid, QStringList users,
                            QStringList groups, QString *ownerUser, QString
                            *ownerGroup)

void setLogicallyVisible (bool v)
    Sets if this dialog is logically visible (i.e. set visible by the action). .

void setBackground (bool v)
    Sets if this dialog is currently in background mode (i.e. not visible). .

void setForceForeground (bool v)
    Sets if this dialog is currently forced to be visible in foreground. .

bool isLogicallyVisible ()
    Returns if this dialog is logically visible (i.e. set visible by the action).

bool isBackground ()
    Returns if this dialog is currently in background mode (i.e. not visible).

QString simpleInputBox (QString text, QString deflt, int valuePreselectFrom = -1, int val-
                        uePreselectTo = -1)

int simpleMessageBox (QString text, int buttons = (int)MessageBoxButton::OK, QString
                     icon = QString(), int defaultbutton = (MessageBoxButton)0, int
                     cancelbutton = (MessageBoxButton)0)

```

### Private Functions

void **\_computevisibility** ()

### Private Members

*Ui::QtActionExecutionInfoDialog* \***ui**

*sh::ui::qt::feedbackpanels::FeedbackPanel* \***currentFeedbackPanel** = 0

### Private Slots

void **on\_btnCancel\_clicked** ()

void **on\_btnBackground\_clicked** ()

**class QtActionExecutionInfoPanel** : public QWidget, public *sh::ui::ActionExecutionInfoPanel*  
#include <qtactionexecutioninfopanel.h> Qt status bar info-panel for an action execution.

### Public Functions

**QtActionExecutionInfoPanel** (*sh::actions::ActionExecutionInfo* \**info*, QColor *color*,  
QWidget \**parent* = 0)

void **setLabel** (QString *s*)  
Sets the label text. .

void **setWidth** (int *width*)  
Sets the panel with (in pixels). .

QSize **sizeHint** () **const**

void **setProgress** (bool *isProgressDeterminate*, quint64 *progressDone*, quint64 *progressAll*)  
Sets the progress. .

void **setForceForeground** (bool *v*)  
Sets if the associated action is currently forced to be visible in foreground. .

void **setPanelVisible** (bool *v*)  
Sets if the panel is visible. .

void **onClicked** (std::function<void>  
> *fcn*QObject \**owner* = 0)Sets a handler for a click on the button (optionally bound to an owner lifetime).

void **onDestroyed** (std::function<void>  
> *fcn*QObject \**owner* = 0)Sets a handler for panel removal (optionally bound to an owner lifetime).

void **onVisibilityChanged** (std::function<void>  
> *fcn*QObject \**owner* = 0)Sets a handler for panel visibility changes (optionally bound to an owner lifetime).



## Private Functions

```
void _check_indeterminateanimationtimer_enable ()
```

## Private Members

```
QString lb11
```

```
bool _progressDeterminate = false
```

```
int _indeterminateAnimationCounter = 0
```

```
int _width
```

```
QTimer *_indeterminateAnimationTimer
```

```
quint64 _progressAll
```

```
quint64 _progressDone
```

```
bool _isforeground_forced = false
```

```
sh::actions::ActionExecutionInfo *info
```

```
QPixmap gradient
```

```
QColor colorProgress2
```

```
QColor colorProgress1
```

```
QColor colorBase
```

```
QColor colorFrame
```

```
QColor colorText
```

```
QColor colorTextDark
```

```
class QtActionMenu : public QMenu
```

```
#include <qtactionmenu.h> Qt based action menu used for context menus and in the toolbar.
```

## Public Functions

```
QtActionMenu (QWidget *parent = 0)
```

```
void setHeader (QString t)
```

```
QList<QAction*> allActions ()
```

```
void clearAllActions ()
```

```
void removeActionAt (int i)
```

```
QAction *addNewAction (int i = -1)
```

```
QAction *addNewSubMenu (sh::ui::qt::QtActionMenu **qsubmenu, int i = -1)
```

```
QAction *addNewSeparator (int i = -1)
```

```
QAction *addNewHeader (int i = -1)
```

```
bool actionIsHeader (QAction *a)
```

### Private Functions

```
void _observeAction (QAction *a)
void _correctMenuPosition ()
```

### Private Members

```
QColor brandingcolor
QFont _boldfont
QFont _normalfont
int _origMenuPosX = -1
int _origMenuPosY = -1
QString _header
int _cntInternalActions
```

### class QActionMenuHandler

*#include <qtactionmenuhandler.h>* A handler which reflects the *sh::actions* objects to a graphical menu (and keeps that up-to-date).

### Public Functions

```
QActionMenuHandler (std::shared_ptr<sh::actions::ActionInstantiation> ai,
                    std::shared_ptr<QtActionMenu> menu)
```

### Private Functions

```
void _markDefault2 ()
void _append_actions (sh::ui::qt::QtActionMenu *menu,
                     QList<std::shared_ptr<sh::actions::ActionInstantiation>> acts,
                     std::shared_ptr<sh::actions::ActionCategory> category)
```

### Private Members

```
QList<std::shared_ptr<sh::actions::ActionInstantiation>> selacts
QList<std::shared_ptr<sh::actions::ActionInstantiation>> diracts
QHash<QAction*, std::shared_ptr<sh::actions::AbstractActionItem>> qaction2action
std::weak_ptr<QtActionMenu> menu
```

## Private Static Functions

```
std::shared_ptr<sh::actions::ActionActionItem> _getDefaultAction (QList<std::shared_ptr<sh::actions::AbstractActionItem>
                                                                actionList)

void _markDefault (std::weak_ptr<sh::actions::SubmenuItem> itmSubmenu,
                  sh::ui::qt::QtActionMenu *menu)

QAction *_createAndConnectAction (sh::actions::AbstractActionItem *itm,
                                  sh::ui::qt::QtActionMenu *menu,
                                  std::function<void>
                                  > onChangedQObject *onChangedbuddy = 0)

void _applyPropertiesToQAction (sh::actions::AbstractActionItem *itm, QAction
                               *_widgetaction)

void _updateSubmenu (sh::actions::SubmenuItem *itm, QtActionMenu *menu,
                    std::function<void>
                    > onChanged = 0)
```

## Friends

```
friend class QtToolbarButtonHandler
```

```
class QtDialog : public QDialog
#include <qtdialog.h> Abstract base class for a qt based dialog. Typically used for also implementing some sh::ui::Dialog.

Subclassed by sh::ui::qt::QtAboutDialog, sh::ui::qt::QtExceptionDialog,
sh::ui::qt::QtFilePropertyDialog, sh::ui::qt::QtLogViewDialog,
sh::ui::qt::QtManageBookmarksDialog, sh::ui::qt::QtManageProfilesDialog,
sh::ui::qt::QtOpenWithDialog, sh::ui::qt::QtStoreProfileDialog, sh::ui::qt::QtTuningDialog
```

## Public Functions

```
QtDialog()
```

```
class QtDialogManager : public sh::ui::DialogManager
#include <qtdialog.h> A DialogManager used in Qt ui.
```

## Public Functions

```
std::shared_ptr<Dialog> getDialogById (qint64 id)
Returns a Dialog by dialog id.
Must be called in main thread.

QList<std::shared_ptr<Dialog>> getAllDialogs ()
Returns a list of all dialogs currently shown (in order of creation).
Must be called in main thread.

QList<qint64> getAllDialogIds ()
Returns a list of dialog ids of all dialogs currently shown (in order of creation).
Must be called in main thread.

template<class TDlg, typename ...Args>
```

`std::shared_ptr<TDlg> createAndShowDialog (Args... args)`

Creates and shows a *Dialog* by class and constructor parameters.

Those dialogs (TDlg) have to implement *Dialog* (typically a subclass of it, representing a particular dialog, like *FilePropertyDialog*), and also some ui mode (e.g. qt, web) specific class (depends on that ui mode).

You typically should not have to use it outside of *MainWindow* or subclasses. The *MainWindow* interface allows to create all available dialogs in a ui mode independent way.

May be called in any thread.

**Return** The created *Dialog* instance.

## Private Functions

`void showDialog (std::shared_ptr<Dialog> dialog) override`

This method implements the ui mode specific mechanism for showing a dialog.

Must be called in main thread.

`class QtExceptionDialog : public sh::ui::qt::QtDialog, public sh::ui::ExceptionDialog`  
`#include <qtexceptiondialog.h>` Qt based exception dialog.

## Public Functions

`QtExceptionDialog (QString error1, QString error2, QString details, QString icon, bool mayRetry, bool mayClose, bool mayCancel, bool showLoglabelAndDetails)`

`~QtExceptionDialog ()`

`ExceptionDialogResult answer () override`

Returns the answer the user has given in this dialog.

`QString error1 ()`

Returns the 1st error text.

`QString error2 ()`

Returns the 2nd error text.

`QString details ()`

Returns the error details.

`QString icon ()`

Returns the icon (as name resolveable by *sh::base::IconManager*).

`bool mayRetry ()`

Returns if it's allowed to retry.

`bool mayClose ()`

Returns if it's allowed to just close and continue without closing Shallot.

`bool mayCancel ()`

Returns if it's allowed to cancel, i.e. throwing a *sh::exceptions::CancelException*.

`bool showLoglabelAndDetails ()`

Returns if the dialog shall allow to read the details.

qint64 **dialogId** ()

Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

bool **isInit** ()

Returns if this dialog is initialized.

Must be called in main thread.

bool **wasAccepted** ()

Returns if this dialog was 'accepted' by the user (typically by clicking some 'OK' button).

Must be called in main thread.

void **waitClosed** ()

Wait until the user closed the dialog in some way.

May be called in any thread.

void **close** ()

Closes the dialog.

Must be called in main thread.

bool **wasClosed** ()

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \***manager** ()

Returns the *DialogManager* which hosts this dialog.

## Private Members

*Ui::QtExceptionDialog* \***ui**

*ExceptionDialogResult* **\_answer** = *ExceptionDialogResult::Shutdown*

## Private Slots

void **on\_btnExit\_clicked** ()

void **on\_btnClose\_clicked** ()

void **on\_btnCancel\_clicked** ()

void **on\_btnRetry\_clicked** ()

void **on\_btnShowLog\_clicked** ()

void **on\_btnExitAndSaveLog\_clicked** ()

void **on\_btnShowDetails\_toggled** (bool *checked*)

**class QtFileDetailsPanel : public QWidget**

*#include <qtfiledetailspanel.h>* The details panel.

Can be shown in the main window. It presents detail information about the selected file.

### Public Functions

```
QtFileDetailsPanel (QWidget *parent = 0)
void setNodes (QList<std::shared_ptr<sh::filesystem::FilesystemNode>> nodes)
void setOrientation (Qt::Orientation orientation)
QSize sizeHint () const override
~QtFileDetailsPanel ()
```

### Private Members

```
int PADDINGX
int PADDINGY
QList<std::shared_ptr<sh::filesystem::FilesystemNode>> _nodes
QList<std::shared_ptr<sh::paneldetails::PanelDetail>> _panelDetails
QList<DetailPlacement> _placements
QFont fontNormal
QFont fontBold
QPixmap _widgetimagecache
QTimer _widgetimagecachetimer
Qt::Orientation _orientation = Qt::Horizontal
class DetailPlacement
    #include <qtfiledetailspanel.h>
```

### Public Functions

```
DetailPlacement (int x, int y, int w, int h, QList<DetailRowPlacement> rowplacements, std::shared_ptr<sh::paneldetails::PanelDetail> detail)
```

### Public Members

```
std::shared_ptr<sh::paneldetails::PanelDetail> detail
int x
int y
int w
int h
QList<DetailRowPlacement> rowplacements
class DetailRowPlacement
    #include <qtfiledetailspanel.h>
```

## Public Functions

**DetailRowPlacement** (int *h*, int *contentx*, int *contenty*, QList<int> *elementwidths*)

**DetailRowPlacement** () = default

**DetailRowPlacement** (const *DetailRowPlacement*&) = default

## Public Members

int **h**

int **contentx**

int **contenty**

QList<int> **elementwidths**

**class QtFileIconview**: public QListView, public *sh::ui::qt::QtFileView*  
*#include <qfileiconview.h>* Icon view for the contents of one directory.

## Public Functions

**QtFileIconview** (QWidget *\*parent* = 0)

Constructed only by the infrastructure and made available otherwise.

void **setThumbDimension** (double *size*)

Sets the size of the thumbnail image.

void **setBackgroundColor** (QString *c*)

QString **backgroundColor** ()

QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> **selectedNodes** ()

void **setSelection** (const QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>>  
*nodes*)

std::shared\_ptr<*sh::filesystem::FilesystemNode*> **node** ()

QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> **getAllVisibleNodes** ()

void **setSizeFormatting** (*sh::filesystem::SizeFormatting* *v*)

void **gotoDir** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> *n*)

void **setHiddenFilesVisible** (bool *value*)

QThread **\*thread** ()

void **configure** (*sh::ui::qt::QtFileViewControl* *\*viewctrl*)

void **focus** ()

void **setSort** (int *column*, Qt::SortOrder *order*)

QObject **\*as\_qobject** ()

QWidget **\*as\_qwidget** ()

QAbstractItemView **\*as\_qabstractitemview** ()

*sh::ui::qt::QtFileViewControl* **\*control** ()

### Private Members

```
int _deleg_w = 0
int _deleg_h = 0
int _dsx = 0
int _dsy = 0
int _lineheight = 0
```

```
class QtFileList : public QTreeView, public sh::ui::qt::QtFileView
    #include <qtfilelist.h> List view for the contents of one directory.
```

### Public Functions

```
QtFileList (QWidget *parent = 0)
    Constructed only by the infrastructure and made available otherwise.

void setSort (int column, Qt::SortOrder order) override
void setBackgroundColor (QString c)
QString backgroundColor ()
QList<std::shared_ptr<sh::filesystem::FilesystemNode>> selectedNodes ()
void setSelection (const QList<std::shared_ptr<sh::filesystem::FilesystemNode>>
                  nodes)
std::shared_ptr<sh::filesystem::FilesystemNode> node ()
QList<std::shared_ptr<sh::filesystem::FilesystemNode>> getAllVisibleNodes ()
void setSizeFormatting (sh::filesystem::SizeFormatting v)
void gotoDir (std::shared_ptr<sh::filesystem::FilesystemNode> n)
void setHiddenFilesVisible (bool value)
QThread *thread ()
void configure (sh::ui::qt::QtFileViewControl *viewctrl)
void focus ()
QObject *as_qobject ()
QWidget *as_qwidget ()
QAbstractItemView *as_qabstractitemview ()
sh::ui::qt::QtFileViewControl *control ()
```



## Private Functions

QString **getColumnNameForIndex** (int *index*)

void **\_adaptColumnWidth** (int *index*)

## Private Members

bool **\_skip\_slot\_sectionResized** = false

## Private Slots

void **slot\_sectionResized** (int *index*, int *oldsize*, int *newsize*)

## Private Static Attributes

std::shared\_ptr<sh::configuration::ConfigurationValue> **cfgvalFileListIconSize** = sh::configuration::Con

**class QtFilePropertyDialog**: public sh::ui::qt::QtDialog, public sh::ui::FilePropertyDialog  
*#include <qfilepropertydialog.h>* Qt based properties dialog.

## Public Functions

**~QtFilePropertyDialog** ()

**QtFilePropertyDialog** (QList<std::shared\_ptr<sh::filesystem::FilesystemNode>>  
*nodes*)

Constructed only by the infrastructure and made available otherwise.

void **init** (std::shared\_ptr<Dialog> *dialog*) **override**

Executes custom stuff on initialization, e.g. for populating the dialog.

void **refresh** () **override**

Refreshes the dialog content.

sh::ui::FilePropertyDialogTabTableView \***createTabViewTable** () **override**

Creates a new tab view table sub-widget.

sh::ui::FilePropertyDialogTabTextView \***createTabViewText** () **override**

Creates a new tab view text sub-widget.

sh::ui::FilePropertyDialogTabIconTextBannerView \***createTabViewIconTextBanner** ()

**override**

Creates a new tab view icon text banner sub-widget.

QList<std::shared\_ptr<FilePropertyDialogTab>> **tabs** ()

Returns a list of all tabs.

sh::ui::FilePropertyDialogTabActionsView \***widgetAt** (std::shared\_ptr<FilePropertyDialogTab>  
*tab*, int *i*)

Returns the widget from a tab at a certain index.

int **widgetCount** (std::shared\_ptr<FilePropertyDialogTab> *tab*)

Returns the number of widgets in a tab.

qint64 **dialogId** ()

Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

bool **isInit** ()

Returns if this dialog is initialized.

Must be called in main thread.

bool **wasAccepted** ()

Returns if this dialog was 'accepted' by the user (typically by clicking some 'OK' button).

Must be called in main thread.

void **waitClosed** ()

Wait until the user closed the dialog in some way.

May be called in any thread.

void **close** ()

Closes the dialog.

Must be called in main thread.

bool **wasClosed** ()

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \***manager** ()

Returns the *DialogManager* which hosts this dialog.

## Public Slots

void **on\_btnClose\_clicked** ()

void **on\_btnRefresh\_clicked** ()

void **on\_tabWidget\_currentChanged** (int *index*)

void **selectTabByScrollPosition** ()

## Public Static Functions

void **addTabFactory** (int *i*, std::shared\_ptr<*FilePropertyDialogTabFactory*> *factory*)

Adds a *FilePropertyDialogTabFactory* so the Properties dialogs show its content.

See also the REGISTER\_FILEPROPERTYDIALOGTAB macro.

### Parameters

- *i*: An index which controls the display order. Use one of the REGISTER\_FILEPROPERTYDIALOGTAB\_INDEX\_\* values in *FilePropertyDialogTabFactory* as base values.

## Private Members

```

Ui::QtFilePropertyDialog *ui
QList<QWidget*> _internaltabwidgets
QList<QString> _tabheads
int _skip_on_tabWidget_currentChanged = 0
int _skip_selectTabByScrollPosition = 0
QVBoxLayout *scrollAreaLayout

```

## Friends

```

friend class FilePropertyDialogTab

class QtFilePropertyDialogTabActionsView : public QWidget, public sh::ui::FilePropertyDialogTabActionsView
#include <qtfilepropertydialogtabactionsview.h> A tab view which shows a main widget in the
main part and some buttons below.

```

## Public Functions

```

QtFilePropertyDialogTabActionsView (QWidget *parent = 0)
    Is intended to be directly constructed from everywhere.

void setContent (FilePropertyDialogTabViewContent *w)
    Sets the main widget. .

void setButtons (QList<QString> buttons)
    Sets the list of buttons. .

void setVisible (bool v)
    Sets the view visible or hidden. .

FilePropertyDialogTabViewContent *content ()
    The main widget.

QList<QString> buttons ()
    The list of buttons.

void onButtonTriggered (std::function<void> int i
    > fct, QObject *owner = 0) Sets a handler for a click on one of the buttons (optionally bound to
    an owner lifetime).

class QtFilePropertyDialogTabIconTextBannerView : public QWidget, public sh::ui::FilePropertyDialogTabIconTextBannerView
#include <qtfilepropertydialogtabicontextbannerview.h> Qt based FilePropertyDialogTabIconTextBannerView.

```

## Public Functions

**QtFilePropertyDialogTabIconTextBannerView** (QWidget *\*parent* = 0)

void **clear** ()  
Clears the contents.

void **insertIcon** (QIcon *icon*, int *sizePt* = 24)  
Adds an icon.

void **insertText** (QString *text*)  
Adds a text.

## Private Members

QHBoxLayout **\*\_layout**

**class QtFilePropertyDialogTabTableView** : public QTableView, public *sh::ui::FilePropertyDialogTabTableView*  
*#include <qtfilepropertydialogtabtableview.h>* Qt based *FilePropertyDialogTabTableView*.

## Public Types

**typedef** QPair<int, int> **ItemIndex**

## Public Functions

**QtFilePropertyDialogTabTableView** (QWidget *\*parent* = 0)

void **setContent** (QList<QPair<QString, QString>> *content*)  
Sets the content.

QList<*ItemIndex*> **getSelectedItems** ()  
Returns the selected cells.

QVariant **getItemValue** (int *r*, int *c*)  
Returns a value of a cell.

## Private Functions

void **createAndSetModel** ()

**class QtFilePropertyDialogTabTextView** : public QLabel, public *sh::ui::FilePropertyDialogTabTextView*  
*#include <qtfilepropertydialogtabtextview.h>* Qt based *FilePropertyDialogTabTextView*.

## Public Functions

**QtFilePropertyDialogTabTextView** (QWidget *\*parent* = 0)

void **setContent** (QString *content*)  
Sets the content.

**class QtFilesystemPanel** : public QWidget  
*#include <qtfilesystempanel.h>* A splitted horizontal panel of file views.

## Public Functions

```

QtFilesystemPanel (QWidget *parent = 0)
~QtFilesystemPanel ()
void addView (bool reinitialize = false)
void removeView (int i)
sh::ui::FileView *currentView ()
int currentViewIndex ()
int viewsCount ()
sh::ui::FileView *view (int i)
void selectFolder (std::shared_ptr<sh::filesystem::FilesystemNode> folder, int index = -
1)
QList<std::shared_ptr<sh::filesystem::FilesystemNode>> selectedNodes ()
void _emit_viewmodeChanged (int i)
void buildView (std::shared_ptr<sh::ui::qt::QtFileView> list,
sh::ui::qt::QtFileViewControl *viewctrl, bool isNew)
void setCustomWidget (int i, std::shared_ptr<QWidget> w)
std::shared_ptr<sh::ui::qt::QtFileView> viewwidget (int i)

```

## Signals

```

void panelFolderActivated (std::shared_ptr<sh::filesystem::FilesystemNode> folder,
bool realSwitch)
void panelCurrentViewChanged ()
void panelViewmodeChanged (int i)
void panelViewOptionChanged (int i)
void panelViewCountChanged ()
void panelSelectionChanged ()

```

## Private Members

```

Ui::QtFilesystemPanel *ui
std::shared_ptr<sh::ui::qt::QtFileView> activeView
QString activeColor
QString inactiveColor
QList<std::shared_ptr<sh::ui::qt::QtFileView>> _views
QList<QWidget*> _mainviews
QHash<QWidget*, std::shared_ptr<QWidget>> _customwidgets
QList<sh::ui::qt::QtFileViewControl*> _viewcontrols
bool _skip_eventfilter = false

```

## Friends

**friend class** ui::qt::QtMainWindow

**friend class** ui::FileView

**class** **QtFileView**: **public** std::enable\_shared\_from\_this<*QtFileView*>  
#include <qtfileview.h> Abstract base class for views for the contents of one directory.  
Subclassed by *sh::ui::qt::QtFileIconview*, *sh::ui::qt::QtFileList*

## Public Functions

**QtFileView** ()

Constructed only by the infrastructure and made available otherwise.

**~QtFileView** ()

void **setBackgroundColor** (QString c)

QString **backgroundColor** ()

QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> **selectedNodes** ()

void **setSelection** (const QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>>  
nodes)

std::shared\_ptr<*sh::filesystem::FilesystemNode*> **node** ()

QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> **getAllVisibleNodes** ()

void **setSizeFormatting** (*sh::filesystem::SizeFormatting* v)

void **gotoDir** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> n)

void **setHiddenFilesVisible** (bool value)

QThread \***thread** ()

void **configure** (*sh::ui::qt::QtFileViewControl* \*viewctrl)

void **focus** ()

void **setSort** (int column, Qt::SortOrder order)

QObject \***as\_qobject** ()

QWidget \***as\_qwidget** ()

QAbstractItemView \***as\_qabstractitemview** ()

*sh::ui::qt::QtFileViewControl* \***control** ()

## Private Members

*sh::ui::qt::QtFileViewControl* \***\_viewctrl** = nullptr

QString **\_tmp\_bgColor**

QPoint **\_dragStartPosition**

int **\_async\_gotodir\_index** = 0

bool **\_dragIsValid** = false

int **\_sort\_column** = 0

```
Qt::SortOrder _sort_order = Qt::AscendingOrder
```

## Friends

```
friend class sh::actions::common::ActionHistoryNavigateForward
friend class sh::actions::common::ActionHistoryNavigateBackward
friend class sh::actions::common::ActionNavigateInHistory
friend class sh::ui::qt::QtFileViewControl
class EventFilter : public QObject
```

## Public Functions

```
EventFilter (QObject *parent, QtFileView *fileview, sh::ui::qt::QtFileViewControl
               *fileviewctrl)
bool eventFilter (QObject *object, QEvent *event)
```

## Private Members

```
QtFileView *fileview
sh::ui::qt::QtFileViewControl *fileviewctrl
class QtFileViewControl : public sh::ui::FileView
    #include <qtfileviewcontrol.h> Qt based file view.
```

## Public Functions

```
QtFileViewControl (sh::ui::qt::QtFilesystemPanel *panel, int i)
~QtFileViewControl ()
sh::ui::ColumnDimensions *columnDimensions () override
    Returns the column dimensions handler. .
sh::tools::HistoryTracker<std::shared_ptr<const sh::filesystem::Eurl>> *historyTracker ()
                                                                    override
    Returns the history tracker. .
FileViewMode viewmode () override
    Returns the view mode (icons, list, ... ?). .
void setViewmode (FileViewMode m) override
    Sets the view mode. .
int index () override
    Returns the position of this file view within the panel. .
void gotoDir (std::shared_ptr<sh::filesystem::FilesystemNode> n) override
    Jumps to a new current directory.
    Implementations have to call the base class implementation inside.
sh::filesystem::SizeFormatting getSizeFormattingMode () override
    Returns the mode how file sizes are formatted for displaying. .
```

```
void setSizeFormattingMode (sh::filesystem::SizeFormatting mode) override
    Sets the mode how file sizes are formatted for displaying. .

bool hiddenFilesVisible () override
    Returns if hidden files are visible. .

void setHiddenFilesVisible (bool v) override
    Sets the visibility of hidden files. .

int iconDimension () override
    Returns the icon size (in pixels). .

void setIconDimension (int v) override
    Sets the icon size (in pixels). .

void setSort (int column, Qt::SortOrder order) override
    Sets how to sort this view. .

int sortColumn () override
    Returns the index of the current sort column. .

Qt::SortOrder sortOrder () override
    Returns the current sort order. .

QList<std::shared_ptr<sh::filesystem::FilesystemNode>> selectedNodes () override
    Returns the nodes which the user has selected in this fileview. .

void setSelection (const      QList<std::shared_ptr<sh::filesystem::FilesystemNode>>
                    nodes) override
    Sets the node selection of this fileview. .

QList<std::shared_ptr<sh::filesystem::FilesystemNode>> getAllVisibleNodes ()
                                                                    override
    Returns a list of all nodes currently listed in this file view. .

void emit_nodesActivated (QList<std::shared_ptr<sh::filesystem::FilesystemNode>>
                          nodes)

void emit_selectionChanged ()

std::shared_ptr<sh::filesystem::FilesystemNode> node ()
    Returns the current directory.

void reload (bool skipModel = false)
    Reloads the data inside this file view.

sh::filesystem::FilesystemModelFileviewProxy *filemodel ()
    Returns the filesystem model for this view. .

void setFilemodel (sh::filesystem::FilesystemModelFileviewProxy *model)
    Sets the filesystem model for this view. .
```

## Signals

```
void nodesActivated (QList<std::shared_ptr<sh::filesystem::FilesystemNode>> nodes)

void selectionChanged ()
    Triggered when the selection have changed.

void viewOptionChanged ()
    Triggered when some view options have changed.
```



## Private Functions

```
void setIndex (int i)
```

## Private Members

```
sh::ui::qt::QtFilesystemPanel * _panel
```

```
int i
```

```
sh::ui::ColumnDimensions * _columnDimensions
```

```
sh::tools::HistoryTracker<std::shared_ptr<const sh::filesystem::Eurl>> * _historyTracker
```

```
FileViewMode _viewmode = FileViewMode::ListView
```

```
bool _hiddenfilesvisible = false
```

```
int _iconDimension = 0
```

```
int _sortColumn
```

```
Qt::SortOrder _sortOrder
```

```
sh::filesystem::SizeFormatting _sizeformattingmode = sh::filesystem::SizeFormatting::SizeFormattingModeF
```

## Friends

```
friend class sh::ui::qt::QtFilesystemPanel
```

```
class QtJumpBar : public QWidget
```

```
#include <qtjumpbar.h> Button bar for navigating to places with parent-buttons and an text entry.
```

## Public Functions

```
QtJumpBar (QWidget *parent = 0)
```

```
~QtJumpBar ()
```

```
void setButtonMode ()
```

```
void setTextMode ()
```

```
bool isTextMode ()
```

```
std::shared_ptr<sh::filesystem::FilesystemNode> node ()
```

```
void setNode (std::shared_ptr<sh::filesystem::FilesystemNode> node)
```

```
QSize sizeHint () const
```

### Signals

```
void nodeChanged ()  
void urlRequested (std::shared_ptr<const sh::filesystem::Eurl> url)
```

### Private Functions

```
void _buildButtons ()
```

### Private Members

```
std::shared_ptr<sh::filesystem::FilesystemNode> _node = 0  
Ui::QtJumpBar *ui  
bool _isTextMode
```

### Private Slots

```
void on_btnOk_clicked ()  
void on_btnAbort_clicked ()
```

```
class QtLinkButton : public QToolButton  
    #include <qtlinkbutton.h> QToolButton with different look.  
    Subclassed by sh::ui::qt::QtSearchPanelButton
```

### Public Functions

```
QtLinkButton (QWidget *parent = 0)  
void setSizeByFactor (int f)  
void setMenu (QStringList items, QString menuchangetxt = tr("(change)"))  
void setSelectedMenuItem (int idx)
```

### Signals

```
void menuItemSelected (int idx)
```

### Private Members

```
QString _linkcolor  
QStringList _menuitems  
QString _menuchangetxt
```

```
class QtLogViewDialog : public sh::ui::qt::QtDialog, public sh::ui::LogViewDialog  
    #include <qtlogviewdialog.h> Qt based log view dialog.
```

## Public Functions

**QtLogViewDialog** (QString *headtext*, QString *subheadtxt*, *base::LogSeverity* *minseverity*)

**~QtLogViewDialog** ()

QString **headtext** ()

Returns the header text.

QString **subheadtxt** ()

Returns the 2nd header text.

*sh::base::LogSeverity* **minimumSeverity** ()

Returns the minimum severity.

qint64 **dialogId** ()

Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

bool **isInitied** ()

Returns if this dialog is initialized.

Must be called in main thread.

bool **wasAccepted** ()

Returns if this dialog was ‘accepted’ by the user (typically by clicking some ‘OK’ button).

Must be called in main thread.

void **waitClosed** ()

Wait until the user closed the dialog in some way.

May be called in any thread.

void **close** ()

Closes the dialog.

Must be called in main thread.

bool **wasClosed** ()

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \***manager** ()

Returns the *DialogManager* which hosts this dialog.

## Private Functions

void **\_reloadLog** ()

void **setMinimumSeverity** (*sh::base::LogSeverity* *minseverity*)

## Private Members

*Ui::QtLogViewDialog* \*ui

## Private Slots

void on\_btnClose\_clicked()

void on\_btnSaveToFile\_clicked()

void on\_btnSeverity\_clicked()

**class QtMainWindow**: public QMainWindow, public *sh::ui::MainWindow*  
#include <qtmainwindow.h> The qt main window implementation.

## Public Functions

**QtMainWindow**(QWidget \*parent = 0)

**~QtMainWindow**()

void **initialize**() **override**

Initializes this main window. .

std::shared\_ptr<*DialogManager*> **dialogManager**() **override**

Returns the *DialogManager* which creates and drives *Dialogs* for this main window.

You typically should not need to use it directly.

int **fileViewCount**() **override**

Returns the current number of file views. .

void **addFileView**(bool *reinitialize* = false) **override**

Adds a new file view. .

void **removeFileView**(int *i*) **override**

Removes a file view. .

*sh::ui::FileView* \***currentFileView**() **override**

Returns the file view which is currently active (i.e. focussed). .

*sh::ui::FileView* \***getFileView**(int *i*) **override**

Returns a file view by position index. .

void **jumpToNode**(std::shared\_ptr<*sh::filesystem::FilesystemNode*> *node*) **override**

Let the current view jump to another location. .

void **setTitlePattern**(QString *value*) **override**

Sets the window title pattern. .

void **setTreeVisible**(bool *v*) **override**

Sets the visibility of the directory tree. .

void **setFileDetailsPanelVisible**(bool *v*) **override**

Sets the visibility of the file details panel. .

std::shared\_ptr<*sh::filesystem::FilesystemNode*> **currentDirectory**() **override**

Gets the *sh::filesystem::FilesystemNode* selected in the current view. .

```

std::shared_ptr<sh::ui::ActionExecutionInfoPanel> addInfoPanel (int          position,
                                                             sh::actions::ActionExecutionInfo
                                                             *info, QColor color =
                                                             QColor()) override

    Creates a new action execution panel.

    Let this smart pointer die for removing it.

void jumpbarSetTextMode () override
    Sets the jumpbar to text input mode. .

void jumpbarSetButtonMode () override
    Sets the jumpbar to button mode. .

bool jumpbarIsTextMode () override
    Returns if the jumpbar is in text input mode. .

QString toolbarPosition () override
    Returns the current toolbar position. .

void setToolbarPosition (QString pos) override
    Sets the toolbar position. .

QString detailPanelPosition () override
    Returns the current detail panel position. .

void setDetailPanelPosition (QString pos) override
    Sets the detail panel position. .

std::shared_ptr<AboutDialog> showAboutDialog () override
    Shows and returns an 'About' dialog. .

std::shared_ptr<ManageProfilesDialog> showManageProfilesDialog () override
    Shows and returns a 'Manage Saved Settings' dialog. .

std::shared_ptr<OpenWithDialog> showOpenWithDialog (std::shared_ptr<sh::filesystem::FilesystemNode>
                                                    node,
                                                    QList<std::shared_ptr<sh::tools::filetypes::OpenMe
                                                    openMethods) override

    Shows and returns a 'Open With' dialog. .

std::shared_ptr<StoreProfileDialog> showStoreProfileDialog (std::shared_ptr<const
                                                            sh::filesystem::Eurl>
                                                            eurl) override

    Shows and returns a 'Save Settings' dialog. .

std::shared_ptr<TuningDialog> showTuningDialog () override
    Shows and returns a 'Tuning' dialog. .

std::shared_ptr<LogViewDialog> showLogViewDialog (QString headtext = QString(),
                                                  QString subheadtxt =
                                                  QString(), sh::base::LogSeverity
                                                  minseverity =
                                                  sh::base::LogSeverity::_DEFAULT)
                                                  override

    Shows and returns a 'Log' dialog. .

std::shared_ptr<ManageBookmarksDialog> showManageBookmarksDialog ()
                                                    override

    Shows and returns a 'Manage Bookmarks' dialog. .

```

`std::shared_ptr<FilePropertyDialog> showFilePropertyDialog (QList<std::shared_ptr<sh::filesystem::FileNodes> nodes) override`  
Shows and returns a ‘File Properties’ dialog. .

`std::shared_ptr<ActionExecutionInfoDialog> createActionExecutionInfoDialog (sh::actions::ActionExecutionInfo *info) override`  
Creates an action execution dialog. .

`void handleCloseAppRejected (QString rejectmsg) override`  
React on a rejected application close request (with some feedback message). .

`void handleClose () override`  
Do some cleanup steps just when closing starts. Implementations must call base implementation!

`sh::ui::qt::QtUIStyle *uiStyle ()`  
Gets a *QtUIStyle* instance, which helps for common ui styling.

`void _initialize (sh::base::SingletonInitializer *singletonInitializer)`  
Initializes the main window. For custom initialization logic, see `MainWindow.initialize`. .

`void fileViewsReloadAll ()`  
Reload all content in each file view.

`void onFileViewOptionsChanged (std::function<void> int  
> fct, QObject *owner = 0)` Sets a handler for some view options in a file view changed (optionally bound to an owner lifetime).

`void onCurrentFileViewChanged (std::function<void>  
> fct, QObject *owner = 0)` Sets a handler for the currently active file view changed (optionally bound to an owner lifetime).

`void onFileViewCountChanged (std::function<void>  
> fct, QObject *owner = 0)` Sets a handler for the number of file views changed (optionally bound to an owner lifetime).

`void onCurrentDirectoryChanged (std::function<void>  
> fct, QObject *owner = 0)` Sets a handler for the current directory in the active file view changed (optionally bound to an owner lifetime).

`void onGlobalViewOptionsChanged (std::function<void>  
> fct, QObject *owner = 0)` Sets a handler for some global view options changed (optionally bound to an owner lifetime).

`void onCurrentProfileChanged (std::function<void>  
> fct, QObject *owner = 0)` Sets a handler for the current profile changed (optionally bound to an owner lifetime).

`void jumpToEurl (std::shared_ptr<const sh::filesystem::Eurl> eurl)`  
Let the current view jump to another location. .

`QString titlePattern ()`  
Returns the window title pattern.

`void setCurrentProfile (QString profile)`  
Sets the current profile. .

`QString currentProfile ()`  
Returns the current profile.

`void reloadProfile ()`  
Reloads the profile data.

bool **treeVisible** ()  
Returns the visibility of the directory tree.

void **setTreeSticky** (bool v)  
Sets if the directory tree should follow the active file view. .

bool **treeSticky** ()  
Returns if the directory tree follows the active file view.

bool **fileDetailsPanelVisible** ()  
Returns the visibility of the file details panel.

std::shared\_ptr<*sh::ui::ActionExecutionInfoPanel*> **addInfoPanel** (*sh::actions::ActionExecutionInfo* \*info, QColor color = QColor())  
Creates a new action execution panel.  
Let this smart pointer die for removing it.

std::shared\_ptr<*ActionExecutionInfoPanel*> **showErrorPanel** ()  
Shows an error panel.

void **\_closeDirectly** ()  
Directly begin application shutdown without checks.

bool **closeApp** ()  
Check if the application may shut down now, and if so, initiate it.

bool **isCloseable** (QString \*msg = nullptr)  
Check if the application may shut down now.

bool **isClosing** ()  
Returns if the application is currently closing.

## Public Static Functions

bool **tryCreateMainWindow** (*MainWindow* \*&mainWindow)  
*QtMainWindow* \*mainWindow ()

void **setMainWindow** (*MainWindow* \*mainWindow)  
Sets the main window instance. .

bool **isReady** ()  
Returns if this process ui is ready (i.e. is created or runs headless).

bool **runsHeadless** ()  
Returns if this process runs headless, i.e. without any ui, just for a background process.

void **\_closeDirectly** (*sh::base::SingletonInitializer* \*singletonInitializer)

QString **uiMode** ()  
Returns the UI mode (e.g. qt, web).

## Private Functions

```
void _setupGlobalShortcut (std::shared_ptr<sh::actions::AbstractActionItem> action)
void _setupGlobalShortcut_recursive (std::shared_ptr<sh::actions::SubmenuItem>
                                     submenu)
void _addPermanentActionToToolBar (std::shared_ptr<sh::actions::SubmenuItem>
                                   a, bool rightSide = false, bool preventDefault-
                                   Action = false)
void _refreshToolBar (std::shared_ptr<sh::actions::ActionInstantiation>)
void _jumpToNode (std::shared_ptr<sh::filesystem::FilesystemNode> node, int skip)
void _jumpToIndex (QModelIndex idx, int skip)
void _newToStatusbar_helper (sh::ui::qt::QtActionExecutionInfoPanel *w)
void _refresh_detailthumbnailvisibility ()
void _thumbnail_resized ()
void _resizethumbnail ()
void _detailpanel_resized ()
void _resizedetailpanel ()
void requestDetailThumbnail (bool force = true)
void setStatusbarVisibility (bool v)
```

## Private Members

```
Ui::QtMainWindow *ui
sh::ui::qt::QtFilesystemPanel *fspanel
sh::filesystem::FilesystemModelDirectoryTreeProxy *treemodel = 0
sh::ui::qt::QtJumpBar *jumpbar
QTimer _statusbar_timer
int _statusbar_fullheight
int _statusbar_targetheight = 0
QSet<sh::ui::qt::QtActionExecutionInfoPanel*> _statusbar_childs
std::shared_ptr<sh::filesystem::FilesystemNode> _currentDirectory = 0
bool _request_detailthumbnail_skip = false
int _thumbnailwidth = 0
Qt::Orientation _myorientation = Qt::Horizontal
sh::ui::qt::QtUIStyle *_uistyle = 0
QAction *_toolbarLeftRightSplitSeparator
QHash<QString, std::shared_ptr<sh::actions::common::ActionGroups>> actionGroupsActions
QList<QtToolBarButtonHandler*> toolbarButtonHandlers
std::shared_ptr<QtDialogManager> _dialogManager
```



### Private Slots

```
void slot_detailbar_moved ()
void slot_toolbar_moved ()
void slot_statusbartimer ()
void slot_treeview_collapsedexpanded (const QModelIndex &index)
```

### Private Static Attributes

```
QtMainWindow *_qtMainWindow = nullptr
```

### Friends

```
friend class QtToolbarButton
friend class QtFilesystemPanel
class MyFlexibleLabel : public QLabel
    #include <qtmainwindow.h> Needed internally in main window in order to have a label which
    really does not request any sizes.
```

### Public Functions

```
MyFlexibleLabel (QWidget *parent = 0)
QSize sizeHint () const override
class QtManageBookmarksDialog : public sh::ui::qt::QtDialog, public sh::ui::ManageBookmarksDialog
    #include <qtmanagebookmarksdialog.h> Qt based manage bookmarks dialog.
```

### Public Functions

```
QtManageBookmarksDialog ()
~QtManageBookmarksDialog ()
qint64 dialogId ()
    Returns the dialog id.
    Each instance has an id unique in the complete Shallot process lifetime.
    Must be called in main thread.
bool isInitiated ()
    Returns if this dialog is initialized.
    Must be called in main thread.
bool wasAccepted ()
    Returns if this dialog was 'accepted' by the user (typically by clicking some 'OK' button).
    Must be called in main thread.
void waitClosed ()
    Wait until the user closed the dialog in some way.
    May be called in any thread.
```

void **close** ()

Closes the dialog.

Must be called in main thread.

bool **wasClosed** ()

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \***manager** ()

Returns the *DialogManager* which hosts this dialog.

### Private Functions

void **readBookmarks** ()

void **\_refresh\_values** ()

void **\_selectbookmark** (QString *id*)

void **\_selectfolder** (QStringList *name*)

### Private Members

*Ui::QtManageBookmarksDialog* \***ui**

### Private Slots

void **on\_btnClose\_clicked** ()

void **on\_treeWidget\_itemSelectionChanged** ()

void **on\_btnDelete\_clicked** ()

void **on\_edtLabel\_textChanged** (const QString &*arg1*)

void **on\_edtLocation\_textChanged** (const QString &*arg1*)

void **on\_btnStore\_clicked** ()

void **on\_btnNew\_clicked** ()

void **on\_btnUp\_clicked** ()

void **on\_btnDown\_clicked** ()

void **on\_btnMove\_clicked** ()

**class** QtManageProfilesDialog : **public** *sh::ui::qt::QtDialog*, **public** *sh::ui::ManageProfilesDialog*  
*#include <qtmanageprofilesdialog.h>* Qt based manage saved settings dialog.

## Public Functions

**QtManageProfilesDialog** ()

**~QtManageProfilesDialog** ()

qint64 **dialogId** ()

Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

bool **isInitiated** ()

Returns if this dialog is initialized.

Must be called in main thread.

bool **wasAccepted** ()

Returns if this dialog was 'accepted' by the user (typically by clicking some 'OK' button).

Must be called in main thread.

void **waitClosed** ()

Wait until the user closed the dialog in some way.

May be called in any thread.

void **close** ()

Closes the dialog.

Must be called in main thread.

bool **wasClosed** ()

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \***manager** ()

Returns the *DialogManager* which hosts this dialog.

## Private Functions

void **\_createProfileList** ()

void **\_createSettingsList** ()

void **\_createNodesList** ()

## Private Members

*Ui*::QtManageProfilesDialog \***ui**

QStringListModel \***\_model**

*sh::settings::ProfileNode* \***\_currentNode** = 0

QHash<QListWidgetItem\*, QString> **\_nodes**

### Private Slots

```
void on_comboBox_currentTextChanged (const QString &arg1)
void on_btnClose_clicked ()
void on_btnDelNode_clicked ()
void on_btnDelProfile_clicked ()
void slot_listViewactivated (QListWidgetItem*, QListWidgetItem*)
```

```
class QtOpenWithDialog : public sh::ui::qt::QtDialog, public sh::ui::OpenWithDialog
#include <qtpenwithdialog.h> Qt based open with dialog.
```

### Public Functions

```
QtOpenWithDialog (std::shared_ptr<sh::filesystem::FilesystemNode> node,
                  QList<std::shared_ptr<sh::tools::filetypes::OpenMethod>> open-
                  Methods)
~QtOpenWithDialog ()
void setProgramList (QList<std::shared_ptr<sh::tools::filetypes::OpenMethod>> open-
                  Methods)
std::shared_ptr<sh::tools::filetypes::OpenMethod> chosenMethod () override
    Returns the chosen open-method (program for opening the file).
bool rememberForMimeType () override
    Returns if the chosen method is checked to be remembered for the same mime-type in the
    future.
std::shared_ptr<const sh::filesystem::Eurl> rememberForDirectory () override
    If it's checked for the chosen method to be remembered for some subdirectory in the future, it
    returns the Eurl of this subdirectory, otherwise nullptr.
bool rememberForFile () override
    Returns if the chosen method is checked to be remembered for the same file in the future.
std::shared_ptr<sh::filesystem::FilesystemNode> node ()
    Returns the file node which is later to be opened.
QList<std::shared_ptr<sh::tools::filetypes::OpenMethod>> openMethods ()
    Returns the open-methods (programs for opening the file) to present for choice.
qint64 dialogId ()
    Returns the dialog id.
    Each instance has an id unique in the complete Shallot process lifetime.
    Must be called in main thread.
bool isInitiated ()
    Returns if this dialog is initialized.
    Must be called in main thread.
bool wasAccepted ()
    Returns if this dialog was 'accepted' by the user (typically by clicking some 'OK' button).
    Must be called in main thread.
```

void **waitClosed** ()  
 Wait until the user closed the dialog in some way.  
 May be called in any thread.

void **close** ()  
 Closes the dialog.  
 Must be called in main thread.

bool **wasClosed** ()  
 Returns if this dialog was closed.  
 Must be called in main thread.

*DialogManager* \***manager** ()  
 Returns the *DialogManager* which hosts this dialog.

### Private Members

std::shared\_ptr<*sh::tools::filetypes::OpenMethod*> **\_chosenMethod**  
 std::shared\_ptr<const *sh::filesystem::Eurl*> **\_rememberfordirectory**  
 QAbstractItemModel \***\_listmodel**  
*Ui::QtOpenWithDialog* \***ui**

### Private Slots

void **on\_btnCancel\_clicked** ()  
 void **on\_btnOk\_clicked** ()  
 void **on\_btnSelectOther\_clicked** ()  
 void **on\_listView\_activated** (const QModelIndex &*index*)  
 void **on\_btnAnotherAncestor\_clicked** ()  
 void **on\_chkRememberDir\_toggled** (bool *checked*)

**class QtOpenWithDialogModel : public QStringListModel**  
*#include <qopenwithdialog.h>* Used internally in OpenWithDialogModel.

### Public Functions

**QtOpenWithDialogModel** (QList<std::shared\_ptr<*sh::tools::filetypes::OpenMethod*>>  
                           *methods*, QObject \**parent* = 0)  
 QVariant **data** (const QModelIndex &*index*, int *role*) **const**

### Private Members

QList<std::shared\_ptr<*sh::tools::filetypes::OpenMethod*>> **\_methods**

**class QtSearchPanel : public QWidget**  
*#include <qtsearchpanel.h>* Search panel for usage in the Qt main window.

### Public Functions

**QtSearchPanel** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> *node*, QWidget *\*parent* = 0)

**~QtSearchPanel** ()

### Friends

**friend class** QtSearchPanelConfiguration

template<class **T1**, class **T2**>

**class QtSearchPanelAbstractEditor : public *T1*, public *T2***  
*#include <qtsearchpanelconfiguration.h>* Helper for other Qt based search panel editors.

### Public Functions

**QtSearchPanelAbstractEditor** (QWidget *\*parent* = 0)

bool **isWidgetEnabled** ()

void **setWidgetEnabled** (bool *v*)

**class QtSearchPanelButton : public *sh::ui::qt::QtLinkButton*, public *sh::ui::SearchPanelButton***  
*#include <qtsearchpanelconfiguration.h>* Qt based *SearchPanelButton*.

### Public Functions

**QtSearchPanelButton** (QWidget *\*parent* = 0)

void **setButtonText** (QString *txt*)  
Sets the button text. .

void **setMenuSelection** (int *i*)  
Sets the currently selected menu item (for buttons with menus). .

void **setSizeByFactor** (int *f*)

void **setMenu** (QStringList *items*, QString *menuchangetxt* = tr("(change)"))

void **setSelectedItem** (int *idx*)

## Signals

void **menuItemSelected** (int *idx*)

**class** **QtSearchPanelConfiguration**: public *sh::ui::SearchPanelConfiguration*  
*#include <qtsearchpanelconfiguration.h>* Search panel configuration for usage in a Qt ui.

## Public Functions

**QtSearchPanelConfiguration** (*QtSearchPanel* \**owner*, QHBoxLayout \**editors*)

*SearchPanelButton* \***addMenuButton** (QString *text*, QStringList *menu*,  
 std::function<void> int  
 > *onchanged*) Adds and returns a button for a menu. .

*SearchPanelButton* \***addActionButton** (QString *text*, std::function<void>  
 > *action*) Adds and returns a button for an action. .

*SearchPanelTextEditor* \***addTextEditor** ()  
 Adds and returns a text editor. .

*SearchPanelDateTimeEditor* \***addDateTimeEditor** ()  
 Adds and returns a date/time editor. .

*SearchPanelLabelEditor* \***addLabel** ()  
 Adds and returns a label. .

*SearchPanelSpacerEditor* \***addSpacer** ()  
 Adds and returns a spacer. .

*SearchPanelAbstractEditor* \***getEditorAt** (int *i*)  
 Returns the editor widget at a given position. .

void **onDestroyed** (std::function<void>  
 > *fcn*) *QObject* \**owner* = 0 Sets a handler for panel removal (optionally bound to an owner life-time).

## Public Members

*\_SearchPanelConfiguration\_HelperQObject* **myqobject**

## Private Members

*QtSearchPanel* \***owner**

QHBoxLayout \***editors**

**class** **QtSearchPanelDateTimeEditor**: public *sh::ui::qt::QtSearchPanelAbstractEditor*<QDateTimeEdit,  
*#include <qtsearchpanelconfiguration.h>* Qt based *SearchPanelDateTimeEditor*.

### Public Functions

**QtSearchPanelDateTimeEditor** (QWidget \*parent = 0)

QDateTime **datetime** ()

Returns the selected date/time. .

void **setDatetime** (QDateTime s)

Sets the selected date/time.

bool **isWidgetEnabled** ()

void **setWidgetEnabled** (bool v)

**class QtSearchPanelLabelEditor** : public *sh::ui::qt::QtSearchPanelAbstractEditor*<QLabel, *sh::ui::SearchPanelLabelEditor*  
*#include <qtsearchpanelconfiguration.h>* Qt based *SearchPanelLabelEditor*.

### Public Functions

**QtSearchPanelLabelEditor** (QWidget \*parent = 0)

QString **textContent** ()

Returns the text content. .

void **setTextContent** (QString s)

Sets the text content. .

*SearchPanelAbstractEditor* \***focusProxyEditor** ()

Returns the focus proxy widget (which focus gets redirected to in some situations). .

void **setFocusProxyEditor** (*SearchPanelAbstractEditor* \*w)

Sets the focus proxy widget (which focus gets redirected to in some situations). .

bool **isWidgetEnabled** ()

void **setWidgetEnabled** (bool v)

**class QtSearchPanelSpacerEditor** : public *sh::ui::qt::QtSearchPanelAbstractEditor*<QLabel, *sh::ui::SearchPanelSpacerEditor*  
*#include <qtsearchpanelconfiguration.h>* Qt based *SearchPanelSpacerEditor*.

### Public Functions

**QtSearchPanelSpacerEditor** (QWidget \*parent = 0)

bool **isWidgetEnabled** ()

void **setWidgetEnabled** (bool v)

**class QtSearchPanelTextEditor** : public *sh::ui::qt::QtSearchPanelAbstractEditor*<QLineEdit, *sh::ui::SearchPanelTextEditor*  
*#include <qtsearchpanelconfiguration.h>* Qt based *SearchPanelTextEditor*.



## Public Functions

**QtSearchPanelTextEditor** (QWidget *\*parent* = 0)

QString **textContent** ()  
Returns the text content. .

void **setTextContent** (QString *s*)  
Sets the text content. .

QString **placeholderDescription** ()  
Returns the placeholder description text (visible if field is empty). .

void **setPlaceholderDescription** (QString *s*)  
Sets the placeholder description text (visible if field is empty). .

bool **isWidgetEnabled** ()

void **setWidgetEnabled** (bool *v*)

**class QtSettingUIFrame**: public QFrame  
*#include <qtsettinguiframe.h>* User interface for one handling one setting.

## Public Functions

**QtSettingUIFrame** (*sh::settings::Setting* *\*setting*, bool *withcheckbox*, QString *display-value*, QString *additionalCheck*, QVariant *additionalCheckValue*, QWidget *\*parent* = 0)

bool **isChecked** ()

void **setChecked** (bool *val*)

void **setAdditionalCheckVisible** (bool *v*)

QVariant **additionalCheckValue** ()

## Private Members

QCheckBox **\_checkbox** = 0

QCheckBox **\_additionalcheckbox** = 0

QLabel **\_additionalcheckboxlabel** = 0

QWidget **\_additionalcheckwidget** = 0

QVariant **\_additionalCheckValue**

**class QtStoreProfileDialog**: public *sh::ui::qt::QtDialog*, public *sh::ui::StoreProfileDialog*  
*#include <qtstoreprofiledialog.h>* Qt based save settings dialog.

## Public Functions

**QtStoreProfileDialog** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*)  
**~QtStoreProfileDialog** ()

**QList<*sh::ui::StoreProfileDialog::SettingEntry*> checkedSettings** () **override**  
Returns the list of settings the user has checked to store.

**QString profileName** () **override**  
Returns the profile the user has chosen to store settings for.

**bool withSubDirectories** () **override**  
Returns if the user has chosen to apply the checked settings also for subdirectories.

**QStringList inheritsFrom** () **override**  
Returns the list of profiles the user has chosen to inherit from.

**bool hasGlobal** () **override**  
Returns if the user has chosen to apply the checked settings for each directory (instead of the current directory).

**std::shared\_ptr<const *sh::filesystem::Eurl*> eurl** ()  
Returns the current directory this dialog shall work on.

**qint64 dialogId** ()  
Returns the dialog id.  
  
Each instance has an id unique in the complete Shallot process lifetime.  
  
Must be called in main thread.

**bool isInitd** ()  
Returns if this dialog is initialized.  
  
Must be called in main thread.

**bool wasAccepted** ()  
Returns if this dialog was ‘accepted’ by the user (typically by clicking some ‘OK’ button).  
  
Must be called in main thread.

**void waitClosed** ()  
Wait until the user closed the dialog in some way.  
  
May be called in any thread.

**void close** ()  
Closes the dialog.  
  
Must be called in main thread.

**bool wasClosed** ()  
Returns if this dialog was closed.  
  
Must be called in main thread.

***DialogManager* \*manager** ()  
Returns the *DialogManager* which hosts this dialog.

### Private Functions

```
void _settingsLevel (int level, bool persist = true)
void _setglobal (bool v)
```

### Private Members

```
QHash<sh::ui::qt::QtSettingUIFrame*, std::shared_ptr<sh::settings::Setting>> _widgets
QHash<int, QLabel*> _headerlabels
QStringList _inheritProfileNameList
bool _global = false
Ui::QtStoreProfileDialog *ui
int _level
```

### Private Slots

```
void on_btnAddProf_clicked ()
void on_btnGlobal_clicked ()
void on_btn22_clicked ()
void on_toolButton_2_clicked ()
void on_toolButton_clicked ()
```

```
class QtToolBarButton : public QToolButton
    #include <qttoolbarbutton.h> A qt toolbar button implementation with optional submenu.
```

### Public Types

```
enum Position
    Values:
    enumerator LEFT
    enumerator RIGHT
    enumerator TOP
    enumerator BOTTOM
```

### Public Functions

```
QtToolBarButton (QWidget *parent = 0)
void setText (const QString &text)
QString text () const
void setIcon (const QIcon &icon)
QIcon icon ()
QSize sizeHint () const
```

```
void setButtonText (QString v)
QString buttonText ()
void setButtonIcon (QIcon v)
QIcon buttonIcon ()
void setButtonEnabled (bool v)
bool isButtonEnabled ()
bool hasExpander ()
void setSubmenu (QtActionMenu *menu)
QtActionMenu *submenu ()
void setLocation (Position location)
void setClickAction (std::function<void>
    > action)
void unsetClickAction ()
void trigger ()
void openMenu ()
```

### Private Functions

```
void _calcDims ()
void computeArrowAndDividerLineCoordinates ()
```

### Private Members

```
QString _text
QString _text1
QString _textWithoutMnemonic1
QString _text2
QString _textWithoutMnemonic2
QIcon _icon
bool _drawIconOnly = false
bool _hovered = false
bool _arrowhovered = false
sh::ui::qt::QtActionMenu * _menu = 0
std::function<void ()> _clickAction
Position location
int tw1
int tworig1
int tw2
```

```
int tworig2
int twmax
int tworigmax
int th
int thorig
int iw
int ih
int iedim
int ew
int eh
const int textpad = 20
const int opad = 3
int expx1
int expx2
int expy1
int expy2
int xt
int yt1
int yt2
int xi
int yi
QPainterPath arrow
bool _drawmnemonics = false
```

### Private Slots

```
void slot_clicked()
```

### Friends

```
friend class sh::actions::ActionsManager
```

```
class QtToolBarButtonHandler
```

*#include <qttoolbarbuttonhandler.h>* A handler which reflects the *sh::actions* objects to a graphical toolbar button (and keeps that up-to-date).

### Public Functions

```
QtToolBarButtonHandler (std::shared_ptr<sh::actions::SubmenuItem> action,  
                        bool preventDefaultAction, QtToolBarButton *button,  
                        QAction *qaction)  
~QtToolBarButtonHandler ()
```

### Private Functions

```
void __updateSubmenu ()
```

### Private Members

```
std::shared_ptr<sh::actions::SubmenuItem> action  
QtActionMenu menu  
bool preventDefaultAction  
QtToolBarButton *button
```

### Private Static Functions

```
void __applyPropertiesToButton (sh::actions::SubmenuItem *action, QtToolBarButton *button, QAction *qaction)  
class QtTuningDialog: public sh::ui::qt::QtDialog, public sh::ui::TuningDialog  
    #include <qttuningdialog.h> Qt based tuning dialog.
```

### Public Functions

```
QtTuningDialog ()  
~QtTuningDialog ()  
qint64 dialogId ()  
    Returns the dialog id.  
    Each instance has an id unique in the complete Shallot process lifetime.  
    Must be called in main thread.  
bool isInited ()  
    Returns if this dialog is initialized.  
    Must be called in main thread.  
bool wasAccepted ()  
    Returns if this dialog was 'accepted' by the user (typically by clicking some 'OK' button).  
    Must be called in main thread.  
void waitClosed ()  
    Wait until the user closed the dialog in some way.  
    May be called in any thread.
```

void **close** ()

Closes the dialog.

Must be called in main thread.

bool **wasClosed** ()

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \***manager** ()

Returns the *DialogManager* which hosts this dialog.

## Private Types

**typedef** QPair<QString, QWidget\*> **BoxWidgetByDesc**

## Private Functions

QString **\_changedlglabel** (std::shared\_ptr<*sh::configuration::ConfigurationValue*> cv)

void **do\_change** (std::shared\_ptr<*sh::configuration::ConfigurationValue*> cv)

void **filterVisibility** (QString s)

## Private Members

*Ui::QtTuningDialog* \***ui**

QList<*BoxWidgetByDesc*> **boxWidgetsByDesc**

## Private Slots

void **on\_btnCancel\_clicked** ()

void **on\_lineEditSearchInt\_textChanged** (const QString &*arg1*)

void **on\_lineEditSearchExt\_textChanged** (const QString &*arg1*)

void **on\_lineEditSearchBehav\_textChanged** (const QString &*arg1*)

void **on\_lineEditSearchAppear\_textChanged** (const QString &*arg1*)

void **on\_tabWidget\_currentChanged** (int *index*)

**class** **QtUIStyle**

*#include <qtuistyle.h>* Methods for applying the Shallot visible style.

See *sh::ui::qt::QtMainWindow* about how to get an instance.

## Public Functions

**QtUIStyle** (*sh::ui::qt::QtMainWindow \*mainwnd*)  
Constructed only by the infrastructure and made available otherwise.

**QColor windowColor** ()

**QColor backgroundColor** ()

**QColor inactiveBackgroundColor** ()

**QColor windowColorHighlighted** ()

**QColor windowColorHalfHighlighted** ()

**QColor windowColorDarkened** ()

**QColor windowColorLikeTitlebar** ()

**QColor linkColor** ()

**QColor titlebarColor** ()

**QColor foregroundColor** ()

**QColor foregroundColorLighter1** ()

**QColor foregroundColorLighter2** ()

**int fontSizeInPt** (int *r* = 100)

**QColor mix** (float *n1*, QColor *c1*, QColor *c2*)

*Sheet* **createSheet** ()

## Private Members

*sh::ui::qt::QtMainWindow* \*\_mainWindow

**class Sheet**  
*#include <qtuistyle.h>*

## Public Functions

**QString sheet** ()

*Sheet* **customcss** (QString *css*)

*Sheet* **fontsize\_byFactor** (int *f*)

*Sheet* **fontsize\_header** ()

*Sheet* **fontsize\_smaller** ()

*Sheet* **textcolor** (QColor *color*)

*Sheet* **textcolor\_lighter1** ()

*Sheet* **textcolor\_lighter2** ()

*Sheet* **background** (QColor *color*, bool *restrictQWidget* = true)

*Sheet* **background\_highlighted** (bool *restrictQWidget* = true)

*Sheet* **background\_halfhighlighted** (bool *restrictQWidget* = true)



```

Sheet background_darkened (bool restrictQWidget = true)
Sheet background_liketitlebar (bool restrictQWidget = true)
Sheet applyTo (QWidget *widget)
Sheet bold ()

```

### Private Functions

```
Sheet (QtUIStyle *style, QString sheet)
```

### Private Members

```

QString _sheet
QtUIStyle *_style

```

### Friends

```
friend class QtUIStyle
```

### namespace feedbackpanels

Qt user interface parts for user feedback in action execution dialogs.

Implementations of *sh::ui::qt::feedbackpanels::FeedbackPanel* are used for implementing parts of *sh::ui::ActionExecutionInfoDialog*.

### Typedefs

```
typedef QPair<QString, QWidget*> GridFormRow
```

```
class Credentials: public sh::ui::qt::feedbackpanels::FeedbackPanel
#include <credentials.h> FeedbackPanel for user login credentials (password based).
```

### Public Functions

```

Credentials (QString domain, QString username, QString password, QString text,
             bool showDomain, bool showUsername, bool showPassword, bool
             showAnonymous, bool showRemember)
~Credentials ()
void cancelRequested ()
bool isClosed ()
void waitUntilClosed ()
int preferredHeight ()

```

### Public Members

QString **domain**  
QString **username**  
QString **password**  
bool **anonymous**  
bool **remember**  
bool **accepted**

### Signals

void **wasClosed**()

### Private Members

*Ui::*Credentials \***ui**

### Private Slots

void **on\_chkAnonymous\_toggled**(bool *checked*)  
void **on\_pushButton\_3\_clicked**()  
void **on\_pushButton\_4\_clicked**()

**class FeedbackPanel** : public QWidget  
    *#include <feedbackpanel.h>* Abstract base class for a Qt helper widget used in action user feedback.

Subclassed by *sh::ui::qt::feedbackpanels::Credentials*, *sh::ui::qt::feedbackpanels::GridForm*, *sh::ui::qt::feedbackpanels::MsgBox*, *sh::ui::qt::feedbackpanels::UnixPermissions*

### Public Functions

**FeedbackPanel**(QWidget \**parent* = 0)  
bool **isClosed**()  
void **waitUntilClosed**()  
int **preferredHeight**()  
void **cancelRequested**() = 0

## Signals

void **wasClosed** ()

## Private Members

QMutex **\_mutex**

QWaitCondition **\_closedcondition**

bool **\_closed** = false

**class GridForm**: public *sh::ui::qt::feedbackpanels::FeedbackPanel*  
#include <gridform.h> *FeedbackPanel* for grid based forms.

## Public Functions

**GridForm**(QString *text*, QList<*sh::ui::qt::feedbackpanels::GridFormRow*> *form*,  
QStringList *answers*, int *defaultanswer* = -1, int *cancelanswer* = -1,  
QWidget *\*parent* = 0)

**~GridForm** ()

int **preferredHeight** ()

void **cancelRequested** ()

bool **isClosed** ()

void **waitUntilClosed** ()

## Public Members

int **answer** = -1

## Signals

void **wasClosed** ()

## Private Members

*Ui::GridForm* **\*ui**

QHash<QPushButton\*, int> **btn2index**

QPushButton **\*\_cancelBtn** = 0

QPushButton **\*\_defaultBtn** = 0

### Private Slots

```
void slot_btnclicked()
```

```
class GridFormInnerSimpleChooser : public QWidget
#include <gridforminnersimplechooser.h> Helper widget for a GridForm.
```

### Public Functions

```
GridFormInnerSimpleChooser (sh::actions::ActionExecutionUserFeedback::Choices
                             *choices, QWidget *parent = 0)
```

```
~GridFormInnerSimpleChooser ()
```

### Public Members

```
int answer = -1
```

```
QString text
```

### Private Functions

```
bool eventFilter (QObject *obj, QEvent *event)
```

### Private Members

```
QLineEdit *lineedit
```

```
sh::actions::ActionExecutionUserFeedback::Choices *choices
```

```
QVBoxLayout *mylayout
```

```
sh::actions::ActionExecutionUserFeedback::Choice *currentchoice = 0
```

### Private Slots

```
void slot_chosen (int id)
```

```
void slot_textedited (QString t)
```

```
class MsgBox : public sh::ui::qt::feedbackpanels::FeedbackPanel
#include <msgbox.h> FeedbackPanel for showing a message, some answer buttons, and optionally a text input box.
```

### Public Functions

```
MsgBox (QString question, QList<QString> answers, QString icon = "", QString with-
        InputBox = QString(), bool inputBoxMultiline = false, int defaultanswer = -1,
        int cancelanswer = -1, int valuePreselectFrom = -1, int valuePreselectTo = -1,
        QList<QString> answericons = QList<QString>(), QWidget *parent = 0)
```

```
~MsgBox ()
```

```
void cancelRequested ()
```

```
bool isClosed ()
```

```
void waitUntilClosed()
```

## Public Members

```
int answer = -1
```

```
QString text
```

## Signals

```
void wasClosed()
```

## Private Members

```
Ui::MsgBox *ui
```

```
QHash<QPushButton*, int> btn2index
```

```
QWidget *_initialFocusWidget = 0
```

```
QPushButton *_cancelBtn = 0
```

```
QPushButton *_defaultBtn = 0
```

```
bool _inputBoxMultiline
```

```
int _valuePreselectFrom
```

```
int _valuePreselectTo
```

## Private Slots

```
void slot_btnclicked()
```

```
class UnixPermissions : public sh::ui::qt::feedbackpanels::FeedbackPanel
    #include <unixpermissions.h> FeedbackPanel for setting one set of unix filesystem permis-
    sions.
```

## Public Functions

```
UnixPermissions (QWidget *parent = 0)
```

```
~UnixPermissions ()
```

```
void cancelRequested ()
```

```
void setflags (bool userMayRead, bool userMayWrite, bool userMayExecute, bool
               groupMayRead, bool groupMayWrite, bool groupMayExecute, bool oth-
               ersMayRead, bool othersMayWrite, bool othersMayExecute, bool sticky,
               bool setuid, bool setgid)
```

```
void setusers (QStringList users, QString selecteduser)
```

```
void setgroups (QStringList groups, QString selectedgroup)
```

```
bool isClosed ()
```

```
void waitUntilClosed()
```

```
int preferredHeight ()
```

### Public Members

```
bool accepted
bool userMayRead
bool userMayWrite
bool userMayExecute
bool groupMayRead
bool groupMayWrite
bool groupMayExecute
bool othersMayRead
bool othersMayWrite
bool othersMayExecute
bool sticky
bool setuid
bool setgid
QString user
QString group
```

### Signals

```
void wasClosed ()
```

### Private Members

```
Ui::UnixPermissions *ui
```

### Private Slots

```
void on_pushButton_clicked ()
void on_pushButton_2_clicked ()
```

### namespace web

The web user interface implementation.

This optional UI provides an application running in a web browser.

## Typedefs

```
typedef QPair<QIcon, int> WebIconTextBannerItemIcon
```

```
typedef QPair<WebIconTextBannerItemIcon, QString> WebIconTextBannerItem
```

```
typedef QMap<QString, QString> WebCommandData
```

```
class WebAboutDialog : public QObject, public sh::ui::web::WebDialog, public sh::ui::AboutDialog  
    #include <webaboutdialog.h> Web based about dialog.
```

## Public Functions

```
WebAboutDialog ()
```

```
QVariant dialogProperty (QString dlgPropertyKey, QVariant deflt = QVariant())
```

Returns a dialog property value by key.

```
void setDialogProperty (QString dlgPropertyKey, QVariant value)
```

Sets a dialog property value for a key.

```
QVariantHash dialogResult ()
```

Returns the dialog result as hash.

The browser side of a web dialog can ‘accept’ a dialog while closing by setting a non-empty dialog result. This mechanism allows browser side dialogs to return data to the backend. It’s up to the backend to interpret the content of this hash.

If the dialog was cancelled, or is still open, the returned hash is empty. See also `wasClosed()`.

```
void triggerDialogEvent (QString name, QJsonObject data = QJsonObject())
```

Triggers a dialog event (at the browsers).

They in turn typically fetch some data again in order to refresh the ui.

```
void setCloseableByUser (bool v = true)
```

Sets if this dialog shall be directly closeable by the user by the window decoration. Set to `false` if the user has to react on this dialog by clicking on some controls in the dialog body.

This method must be called during construction.

```
bool isCloseableByUser ()
```

Returns if this dialog shall be directly closeable by the user by the window decoration.

```
QJsonObject toJson () override
```

Returns the json representation as QJsonObject.

```
qint64 dialogId ()
```

Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

```
bool isInit ()
```

Returns if this dialog is initialized.

Must be called in main thread.

```
bool wasAccepted ()
```

Returns if this dialog was ‘accepted’ by the user (typically by clicking some ‘OK’ button).

Must be called in main thread.

void **waitClosed** ()  
Wait until the user closed the dialog in some way.  
May be called in any thread.

void **close** ()  
Closes the dialog.  
Must be called in main thread.

bool **wasClosed** ()  
Returns if this dialog was closed.  
Must be called in main thread.

*DialogManager* \***manager** ()  
Returns the *DialogManager* which hosts this dialog.

**class WebActionExecutionInfoDialog**: public *sh::ui::ActionExecutionInfoDialog*  
*#include <webactionexecutioninfodialog.h>* Web based action execution dialog.

## Public Types

**enum MessageBoxButton**  
Buttons in a message box from *ActionExecutionUserFeedback*.

*Values:*

**enumerator** NONE = 0  
**enumerator** OK = 1 << 0  
**enumerator** Continue = 1 << 1  
**enumerator** Cancel = 1 << 2  
**enumerator** Retry = 1 << 3  
**enumerator** Yes = 1 << 4  
**enumerator** No = 1 << 5

## Public Functions

**WebActionExecutionInfoDialog** (*sh::actions::ActionExecutionInfo* \*info)

void **setDetails** (QString fv, QString fob, QString tv, QString tob)  
Sets the item details text ('from a/foo.jpg', 'to b/foo.jpg'). .

void **setHead** (QString txt)  
Sets the header text. .

void **setProgress** (bool isDeterminate, quint64 done, quint64 all, QString text)  
Sets the progress. .

int **messageBox** (QString text, QList<QString> answers, QString icon = QString(), int defaultanswer = -1, int cancelanswer = -1, QList<QString> answericons = QList<QString>())

int **inputBox** (QString text, QList<QString> answers, QString \*value, QString icon = QString(), int defaultanswer = -1, int cancelanswer = -1, int valuePreselectFrom = -1, int valuePreselectTo = -1)



```

int multilineInputDialog (QString text, QList<QString> answers, QString *value, QString
                           icon = QString(), int defaultanswer = -1, int cancelanswer = -1)

int simpleChooserGridform (QString text, GridformEntries *entries, QStringList an-
                           swers, int defaultanswer = -1, int cancelanswer = -1)

bool credentialsDialog (QString text, bool showDomain, bool showUsername, bool
                        showPassword, bool showAnonymous, bool showRemember,
                        QString *domain, QString *username, QString *password,
                        bool *isAnonymous, bool *isRemember)

bool unixPermissionsDialog (bool *userMayRead, bool *userMayWrite, bool *user-
                            MayExecute, bool *groupMayRead, bool *groupMay-
                            Write, bool *groupMayExecute, bool *othersMayRead,
                            bool *othersMayWrite, bool *othersMayExecute, bool
                            *sticky, bool *setuid, bool *setgid, QStringList users,
                            QStringList groups, QString *ownerUser, QString
                            *ownerGroup)

qint64 id ()
qint64 webts_created ()
QString details_fromverb ()
QString details_fromobject ()
QString details_toverb ()
QString details_toobject ()
QString head ()
bool progress_isDeterminate ()
quint64 progress_done ()
quint64 progress_all ()
QString progress_text ()
QJsonValue userFeedbackAsJsonValue ()
bool isLogicallyVisible ()
    Returns if this dialog is logically visible (i.e. set visible by the action).
void setLogicallyVisible (bool v)
    Sets if this dialog is logically visible (i.e. set visible by the action). .
bool isBackground ()
    Returns if this dialog is currently in background mode (i.e. not visible).
void setBackground (bool v)
    Sets if this dialog is currently in background mode (i.e. not visible). .
void setForceForeground (bool v)
    Sets if this dialog is currently forced to be visible in foreground. .
QString simpleInputDialog (QString text, QString deft, int valuePreselectFrom = -1, int val-
                           uePreselectTo = -1)
int simpleMessageBox (QString text, int buttons = (int) MessageBoxButton::OK, QString
                      icon = QString(), int defaultbutton = (MessageBoxButton)0, int
                      cancelbutton = (MessageBoxButton)0)

```

### Private Functions

```
void _handleUserFeedback (QString kind, std::shared_ptr<UserFeedback> userfeedback)  
void _triggerChanged ()
```

### Private Members

```
qint64 _id  
QString _details_fv  
QString _details_fob  
QString _details_tv  
QString _details_tob  
QString _head  
bool _progress_isDeterminate  
qint64 _progress_done  
qint64 _progress_all  
QString _progress_text  
std::shared_ptr<UserFeedback> _currentUserFeedback = 0  
QMutex _currentUserFeedbackMutex  
QWaitCondition _currentUserFeedbackChangedCondition  
qint64 _webts_created
```

### Private Static Functions

```
QByteArray iconToBase64Src (QString icon, int sizeInPt)  
QList<QVariant> iconsToBase64Srcs (QStringList icons, int sizeInPt)
```

### Friends

```
friend class WebActionManager  
class UserFeedback : public QMap<QString, QVariant>  
    #include <webactionexecutioninfodialog.h>
```

## Public Members

qint64 **id** = -1  
 bool **answered** = false

**class WebActionExecutionInfoPanel** : public *sh::ui::ActionExecutionInfoPanel*  
*#include <webactionexecutioninfopanel.h>* Web based action execution info panel.

## Public Functions

**WebActionExecutionInfoPanel** (*sh::actions::ActionExecutionInfo* \**info*, QColor *color*)  
**~WebActionExecutionInfoPanel** ()  
 void **setLabel** (QString *s*)  
     Sets the label text. .  
 void **setProgress** (bool *isProgressDeterminate*, qint64 *progressDone*, qint64 *progressAll*)  
     Sets the progress. .  
 void **setForceForeground** (bool *v*)  
     Sets if the associated action is currently forced to be visible in foreground. .  
 void **setPanelVisible** (bool *v*)  
     Sets if the panel is visible. .  
 void **setWidth** (int *width*)  
     Sets the panel with (in pixels). .  
 qint64 **id** ()  
 QString **label** ()  
 bool **isProgressDeterminate** ()  
 qint64 **progressDone** ()  
 qint64 **progressAll** ()  
 bool **forceForeground** ()  
 bool **panelVisible** ()  
 int **width** ()  
 QColor **color** ()  
*sh::actions::ActionExecutionInfo* \***info** ()  
 void **onClicked** (std::function<void>  
     > *fcn*QObject \**owner* = 0)Sets a handler for a click on the button (optionally bound to an owner lifetime).  
 void **onDestroyed** (std::function<void>  
     > *fcn*QObject \**owner* = 0)Sets a handler for panel removal (optionally bound to an owner lifetime).  
 void **onVisibilityChanged** (std::function<void>  
     > *fcn*QObject \**owner* = 0)Sets a handler for panel visibility changes (optionally bound to an owner lifetime).

## Private Functions

void **\_triggerChanged** ()

## Private Members

QColor **\_color**

*sh::actions::ActionExecutionInfo* \***\_info**

qint64 **\_id**

QString **\_label**

bool **\_isProgressDeterminate**

qint64 **\_progressDone**

qint64 **\_progressAll**

bool **\_forceForeground**

bool **\_panelVisible**

int **\_width**

## Friends

**friend class** WebActionManager

**class WebActionManager** : public QObject, public *sh::ui::web::WebModule*  
*#include <webactionmanager.h>* Manages *sh::actions::AbstractActionItem* handling, e.g. showing them in the toolbar, executing them, and displaying their user interface.

## Public Functions

**WebActionManager** ()

void **initialize** ()

std::shared\_ptr<*WebActionExecutionInfoPanel*> **addInfoPanel** (int *position*,  
*sh::actions::ActionExecutionInfo*  
\**info*, QColor *color* =  
QColor())

Adds and returns a new action info panel.

std::shared\_ptr<*WebActionExecutionInfoDialog*> **addActionExecutionInfoDialog** (*sh::actions::ActionExecutionInfo*  
\**info*)

Adds and returns a new action info dialog.

void **refreshToolbar** (std::shared\_ptr<*sh::actions::ActionInstantiation*> *instantiation*)  
Refreshes the main toolbar.

bool **rootCommand** (*WebServerEngineRequest* \**request*, QString *webadapter*)  
Returns the root ('index.html') content. .

void **setEngine** (*WebServerEngine* \**webserver*)  
Assigns this web module to a *sh::ui::web::WebServerEngine*. .

*WebServerEngine* \***webserver** ()  
Returns the *sh::ui::web::WebServerEngine* this web module is assigned to (may be nullptr).

## Public Static Functions

```
bool executeCommand_byQObjectReflection (QObject      *o,      Web-
                                         ServerEngineRequest *request,
                                         QString command, QString justLeft-
                                         side = QString())
```

Convenience function often used by *executeCommand()*.

For a QObject, it searches there for a slot with the name cmd\_{command}, with `/'s replaced by '\_'s (so, for the command "foo/get\_bars", it searches for slotcmd\_foo\_get\_bars). If it finds one, it gets executed (in a worker thread). The request is considered as handled, if so.

If there is such a slot, but with appended \_\_M in name, it is called in main thread!

**Return** If it handled (answered) the command.

## Private Functions

```
bool executeCommand (WebServerEngineRequest *request,  QString  command)
                    override
```

Executes command as requested by the browser, if responsible.

Read parameters, check if this module is responsible for answering, and then write an answer with request.

Implementations often use *executeCommand\_byQObjectReflection()* for convenience.

**Return** If it handled (answered) the command.

```
void _observeToolBarAction (sh::actions::AbstractActionItem *a)
```

```
std::shared_ptr<sh::actions::AbstractActionItem> resolveActionpath (QString      sac-
                                                                    tionpath,
                                                                    QStringList
                                                                    *actionpath,
                                                                    QList<QPair<QString,
                                                                    std::shared_ptr<sh::actions::AbstractA
                                                                    *subactions)
```

```
void _triggerActionuiChanged ()
```

## Private Members

```
QList<WebActionExecutionInfoPanel*> _infopanel
```

```
QList<WebActionExecutionInfoDialog*> _infodialogs
```

```
QTimer _triggerActionuiChangedTimer
```

```
QTimer _triggerToolBarRefreshTimer
```

```
QMap<QString, std::shared_ptr<sh::actions::SubmenuActionItem>> _permanentToolBarActions
```

### Private Slots

```
void cmd__answer_userfeedback__M (WebServerEngineRequest *request)
void cmd__execute (WebServerEngineRequest *request)
void cmd__get (WebServerEngineRequest *request)
void cmd__get_ui__M (WebServerEngineRequest *request)
void cmd__icon (WebServerEngineRequest *request)
void cmd__panel_clicked__M (WebServerEngineRequest *request)
```

### Private Static Attributes

QRegularExpression **reActionTextAccel**

### Friends

```
friend class WebActionExecutionInfoPanel
friend class WebActionExecutionInfoDialog
```

```
class WebDetailPanelManager : public QObject, public sh::ui::web::WebModule
#include <webdetailpanelmanager.h> Manages the detail panel (typically in bottom part of main
window).
```

### Public Functions

**WebDetailPanelManager** ()

void **load** (QList<std::shared\_ptr<sh::filesystem::FilesystemNode>> nodes)  
Loads details for a given list of nodes (typically called after they get selected).

bool **rootCommand** (WebServerEngineRequest \*request, QString webadapter)  
Returns the root ('index.html') content. .

void **setEngine** (WebServerEngine \*webserver)  
Assigns this web module to a *sh::ui::web::WebServerEngine*. .

*WebServerEngine* \***webserver** ()  
Returns the *sh::ui::web::WebServerEngine* this web module is assigned to (may be nullptr).

### Public Static Functions

```
bool executeCommand_byQObjectReflection (QObject *o, Web-
ServerEngineRequest *request,
QString command, QString justLeft-
side = QString())
```

Convenience function often used by *executeCommand()*.

For a QObject, it searches there for a slot with the name `cmd_{command}`, with ``'s replaced by '\_s (so, for the command "foo/get\_bars", it searches for `slotcmd_foo_get_bars``). If it finds one, it gets executed (in a worker thread). The request is considered as handled, if so.

If there is such a slot, but with appended `__M` in name, it is called in main thread!

**Return** If it handled (answered) the command.

## Private Functions

```
bool executeCommand (WebServerEngineRequest *request, QString command)
    override
```

Executes `command` as requested by the browser, if responsible.

Read parameters, check if this module is responsible for answering, and then write an answer with `request`.

Implementations often use `executeCommand_byQObjectReflection()` for convenience.

**Return** If it handled (answered) the command.

## Private Members

QMutex **\_mutex**

QList<std::shared\_ptr<*sh::paneldetails::PanelDetail*>> **\_paneldetails**

qint64 **\_webts\_lastupdate**

## Private Slots

```
void cmd__get_details (WebServerEngineRequest *request)
```

```
void cmd__get_thumbnail (WebServerEngineRequest *request)
```

```
void cmd__link_clicked (WebServerEngineRequest *request)
```

```
class WebDialog: public sh::tools::Jsonable
```

`#include <webdialog.h>` Abstract base class for a web based dialog. Typically used for also implementing some *sh::ui::Dialog*.

A web dialog is a dialog window presented to the user on browser side. A web dialog implementation consists of a backend part (a subclass of *WebDialog*) and a browser side part (basically a subclass of *shwebui.Dialog*).

For implementing a web dialog, implement a subclass of *WebDialog* (which in turn will specify the browser side part) and *sh::ui::Dialog*. Create such dialogs by means of *WebDialogManager*.

Communicating values between backend and browser typically works by *dialogProperty()* and *setDialogProperty()*, also *dialogResult()* for the dialog result after closing.

Subclassed by *sh::ui::web::WebAboutDialog*, *sh::ui::web::WebExceptionDialog*,  
*sh::ui::web::WebFilePropertyDialog*, *sh::ui::web::WebLogViewDialog*,  
*sh::ui::web::WebManageBookmarksDialog*, *sh::ui::web::WebManageProfilesDialog*,  
*sh::ui::web::WebOpenWithDialog*, *sh::ui::web::WebStoreProfileDialog*,  
*sh::ui::web::WebTuningDialog*

## Public Functions

**WebDialog** (QString *dialogClassName*)

See [WebDialogManager](#).

### Parameters

- **dialogClassName**: The name of the shwebui.Dialog subclass which implements the web dialog on browser side.

QVariant **dialogProperty** (QString *dlgPropertyKey*, QVariant *deflt* = QVariant())

Returns a dialog property value by key.

void **setDialogProperty** (QString *dlgPropertyKey*, QVariant *value*)

Sets a dialog property value for a key.

QVariantHash **dialogResult** ()

Returns the dialog result as hash.

The browser side of a web dialog can ‘accept’ a dialog while closing by setting a non-empty dialog result. This mechanism allows browser side dialogs to return data to the backend. It’s up to the backend to interpret the content of this hash.

If the dialog was cancelled, or is still open, the returned hash is empty. See also `wasClosed()`.

void **triggerDialogEvent** (QString *name*, QJsonObject *data* = QJsonObject())

Triggers a dialog event (at the browsers).

They in turn typically fetch some data again in order to refresh the ui.

void **setCloseableByUser** (bool *v* = true)

Sets if this dialog shall be directly closeable by the user by the window decoration. Set to `false` if the user has to react on this dialog by clicking on some controls in the dialog body.

This method must be called during construction.

bool **isCloseableByUser** ()

Returns if this dialog shall be directly closeable by the user by the window decoration.

**~WebDialog** ()

QJsonObject **toJson** () **override**

Returns the json representation as QJsonObject.

## Private Members

QVariantHash **\_properties**

QString **\_dialogClassName**

qint64 **\_webts\_created**



## Friends

**friend class** WebDialogManager

**class** WebDialogManager : public QObject, public *sh::ui::web::WebModule*, public *sh::ui::DialogManager*  
*#include <webdialog.h>* A *DialogManager* used in Web ui.

## Public Functions

**WebDialogManager** ()

*WebServerEngine* \***webserver** ()

Return the *WebServerEngine* hosting this dialog.

bool **rootCommand** (*WebServerEngineRequest* \*request, QString webadapter)

Returns the root ('index.html') content. .

void **setEngine** (*WebServerEngine* \*webserver)

Assigns this web module to a *sh::ui::web::WebServerEngine*. .

std::shared\_ptr<*Dialog*> **getDialogById** (qint64 id)

Returns a *Dialog* by dialog id.

Must be called in main thread.

QList<std::shared\_ptr<*Dialog*>> **getAllDialogs** ()

Returns a list of all dialogs currently shown (in order of creation).

Must be called in main thread.

QList<qint64> **getAllDialogIds** ()

Returns a list of dialog ids of all dialogs currently shown (in order of creation).

Must be called in main thread.

template<class **TDlg**, typename ...**Args**>

std::shared\_ptr<*TDlg*> **createAndShowDialog** (*Args... args*)

Creates and shows a *Dialog* by class and constructor parameters.

Those dialogs (TDlg) have to implement *Dialog* (typically a subclass of it, representing a particular dialog, like *FilePropertyDialog*), and also some ui mode (e.g. qt, web) specific class (depends on that ui mode).

You typically should not have to use it outside of *MainWindow* or subclasses. The *MainWindow* interface allows to create all available dialogs in a ui mode independent way.

May be called in any thread.

**Return** The created *Dialog* instance.

## Public Static Functions

bool **executeCommand\_byQObjectReflection** (QObject \*o, *WebServerEngineRequest* \*request, QString command, QString justLeftSide = QString())

Convenience function often used by *executeCommand()*.

For a QObject, it searches there for a slot with the name cmd\_{command}, with ``'`'s replaced by '\_'s (so, for the command "foo/getBars", it searches for

slotcmd\_foo\_get\_bars`). If it finds one, it gets executed (in a worker thread). The request is considered as handled, if so.

If there is such a slot, but with appended `__M` in name, it is called in main thread!

**Return** If it handled (answered) the command.

### Private Functions

std::shared\_ptr<Dialog> **getDialogForRequest** (*WebServerEngineRequest* \*request)

Returns the *WebDialog* referred to the request (i.e. its dialogId parameter).

bool **executeCommand** (*WebServerEngineRequest* \*request, QString command)  
**override**

Executes command as requested by the browser, if responsible.

Read parameters, check if this module is responsible for answering, and then write an answer with request.

Implementations often use *executeCommand\_byQObjectReflection()* for convenience.

**Return** If it handled (answered) the command.

void **showDialog** (std::shared\_ptr<Dialog> dialog) **override**

This method implements the ui mode specific mechanism for showing a dialog.

Must be called in main thread.

### Private Slots

void **cmd\_change\_dialog\_property\_\_M** (*WebServerEngineRequest* \*request)

void **cmd\_closed\_in\_browser\_\_M** (*WebServerEngineRequest* \*request)

void **cmd\_list\_\_M** (*WebServerEngineRequest* \*request)

```
class WebExceptionDialog : public QObject, public sh::ui::web::WebDialog, public sh::ui::ExceptionDialog {
    #include <webexceptiondialog.h> Web based exception dialog.
```

### Public Functions

**WebExceptionDialog** (QString error1, QString error2, QString details, QString icon, bool mayRetry, bool mayClose, bool mayCancel, bool showLoglabelAndDetails)

*ExceptionDialogResult* **answer** () **override**

Returns the answer the user has given in this dialog.

QVariant **dialogProperty** (QString dlgPropertyKey, QVariant deflt = QVariant())

Returns a dialog property value by key.

void **setDialogProperty** (QString dlgPropertyKey, QVariant value)

Sets a dialog property value for a key.

QVariantHash **dialogResult** ()

Returns the dialog result as hash.

The browser side of a web dialog can ‘accept’ a dialog while closing by setting a non-empty dialog result. This mechanism allows browser side dialogs to return data to the backend. It’s up to the backend to interpret the content of this hash.

If the dialog was cancelled, or is still open, the returned hash is empty. See also wasClosed().

void **triggerDialogEvent** (QString *name*, QJsonObject *data* = QJsonObject())  
 Triggers a dialog event (at the browsers).

They in turn typically fetch some data again in order to refresh the ui.

void **setCloseableByUser** (bool *v* = true)  
 Sets if this dialog shall be directly closeable by the user by the window decoration. Set to *false* if the user has to react on this dialog by clicking on some controls in the dialog body.

This method must be called during construction.

bool **isCloseableByUser** ()  
 Returns if this dialog shall be directly closeable by the user by the window decoration.

QJsonObject **toJson** () **override**  
 Returns the json representation as QJsonObject.

QString **error1** ()  
 Returns the 1st error text.

QString **error2** ()  
 Returns the 2nd error text.

QString **details** ()  
 Returns the error details.

QString **icon** ()  
 Returns the icon (as name resolveable by *sh::base::IconManager*).

bool **mayRetry** ()  
 Returns if it's allowed to retry.

bool **mayClose** ()  
 Returns if it's allowed to just close and continue without closing Shallot.

bool **mayCancel** ()  
 Returns if it's allowed to cancel, i.e. throwing a *sh::exceptions::CancelException*.

bool **showLoglabelAndDetails** ()  
 Returns if the dialog shall allow to read the details.

qint64 **dialogId** ()  
 Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

bool **isInited** ()  
 Returns if this dialog is initialized.

Must be called in main thread.

bool **wasAccepted** ()  
 Returns if this dialog was 'accepted' by the user (typically by clicking some 'OK' button).

Must be called in main thread.

void **waitClosed** ()  
 Wait until the user closed the dialog in some way.

May be called in any thread.

void **close** ()  
 Closes the dialog.

Must be called in main thread.

bool **wasClosed** ()

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \***manager** ()

Returns the *DialogManager* which hosts this dialog.

### Private Slots

void **cmd\_\_icon\_\_M** (*WebServerEngineRequest* \*request)

**class WebFilePropertyDialog** : public QObject, public *sh::ui::web::WebDialog*, public *sh::ui::FilePr*  
#include <webfilepropertydialog.h> Web based file property dialog.

### Public Functions

void **refresh** () **override**

Refreshes the dialog content.

*FilePropertyDialogTabTableView* \***createTabViewTable** () **override**

Creates a new tab view table sub-widget.

*FilePropertyDialogTabTextView* \***createTabViewText** () **override**

Creates a new tab view text sub-widget.

*FilePropertyDialogTabIconTextBannerView* \***createTabViewIconTextBanner** () **override**

Creates a new tab view icon text banner sub-widget.

**WebFilePropertyDialog** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>>  
nodes)

**~WebFilePropertyDialog** ()

QVariant **dialogProperty** (QString *dlgPropertyKey*, QVariant *deflt* = QVariant())

Returns a dialog property value by key.

void **setDialogProperty** (QString *dlgPropertyKey*, QVariant *value*)

Sets a dialog property value for a key.

QVariantHash **dialogResult** ()

Returns the dialog result as hash.

The browser side of a web dialog can ‘accept’ a dialog while closing by setting a non-empty dialog result. This mechanism allows browser side dialogs to return data to the backend. It’s up to the backend to interpret the content of this hash.

If the dialog was cancelled, or is still open, the returned hash is empty. See also wasClosed().

void **triggerDialogEvent** (QString *name*, QJsonObject *data* = QJsonObject())

Triggers a dialog event (at the browsers).

They in turn typically fetch some data again in order to refresh the ui.

void **setCloseableByUser** (bool *v* = true)

Sets if this dialog shall be directly closeable by the user by the window decoration. Set to *false* if the user has to react on this dialog by clicking on some controls in the dialog body.

This method must be called during construction.

bool **isCloseableByUser** ()

Returns if this dialog shall be directly closeable by the user by the window decoration.

QJsonObject **toJson** () **override**

Returns the json representation as QJsonObject.

QList<std::shared\_ptr<*FilePropertyDialogTab*>> **tabs** ()

Returns a list of all tabs.

*sh::ui::FilePropertyDialogTabActionsView* \***widgetAt** (std::shared\_ptr<*FilePropertyDialogTab*>  
tab, int i)

Returns the widget from a tab at a certain index.

int **widgetCount** (std::shared\_ptr<*FilePropertyDialogTab*> tab)

Returns the number of widgets in a tab.

qint64 **dialogId** ()

Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

bool **isInitiated** ()

Returns if this dialog is initialized.

Must be called in main thread.

bool **wasAccepted** ()

Returns if this dialog was 'accepted' by the user (typically by clicking some 'OK' button).

Must be called in main thread.

void **waitClosed** ()

Wait until the user closed the dialog in some way.

May be called in any thread.

void **close** ()

Closes the dialog.

Must be called in main thread.

bool **wasClosed** ()

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \***manager** ()

Returns the *DialogManager* which hosts this dialog.

## Public Static Functions

void **addTabFactory** (int i, std::shared\_ptr<*FilePropertyDialogTabFactory*> factory)

Adds a *FilePropertyDialogTabFactory* so the Properties dialogs show its content.

See also the REGISTER\_FILEPROPERTYDIALOGTAB macro.

### Parameters

- i: An index which controls the display order. Use one of the REGISTER\_FILEPROPERTYDIALOGTAB\_INDEX\_\* values in *FilePropertyDialogTabFactory* as base values.

### Private Functions

```
void tabwidgetTableSelect (int cookie, int tabidx, int widgetidx, QList<int> lsel)  
void triggerAction (int cookie, int tabidx, int widgetidx, int btnidx)  
QJsonArray getProperties ()
```

### Private Members

```
int _cookie = 0  
QTimer _refreshTimer  
QList<WebFilePropertyDialogChild*> _childs
```

### Private Slots

```
void cmd__get_properties__M (WebServerEngineRequest *request)  
void cmd__refresh__M (WebServerEngineRequest *request)  
void cmd__trigger_action__M (WebServerEngineRequest *request)  
void cmd__tabwidget_table_select__M (WebServerEngineRequest *request)
```

### class WebFilePropertyDialogChild

Subclassed by *sh::ui::web::WebFilePropertyDialog::WebFilePropertyDialogTabActionsView*,  
*sh::ui::web::WebFilePropertyDialog::WebFilePropertyDialogTabIconTextBannerView*,  
*sh::ui::web::WebFilePropertyDialog::WebFilePropertyDialogTabTableView*,  
*sh::ui::web::WebFilePropertyDialog::WebFilePropertyDialogTabTextView*

### Public Functions

```
WebFilePropertyDialogChild (WebFilePropertyDialog *dlg)  
~WebFilePropertyDialogChild ()  
class WebFilePropertyDialogTabActionsView : public sh::ui::FilePropertyDialogTabActionsView
```

### Public Functions

```
WebFilePropertyDialogTabActionsView (WebFilePropertyDialog *dlg)  
void setVisible (bool v) override  
    Sets the view visible or hidden. .  
void setContent (FilePropertyDialogTabViewContent *cnt) override  
    Sets the main widget. .  
void setButtons (QList<QString> buttons) override  
    Sets the list of buttons. .  
void setLabelText (QString text)  
QJsonObject toJson () override  
    Returns the json representation as QJsonObject.
```

*FilePropertyDialogTabViewContent* \*content ()

The main widget.

QList<QString> buttons ()

The list of buttons.

void onButtonTriggered (std::function<void> int i

> fct, QObject \*owner = 0) Sets a handler for a click on one of the buttons (optionally bound to an owner lifetime).

## Private Members

bool \_visible = false

QString \_labelText

## Friends

friend class WebFilePropertyDialog

class WebFilePropertyDialogTabIconTextBannerView : public *sh::ui::FilePropertyDialogTab*

## Public Functions

WebFilePropertyDialogTabIconTextBannerView (*WebFilePropertyDialog*  
\*dlg)

void clear () override

Clears the contents.

void insertIcon (QIcon icon, int sizePt = 24) override

Adds an icon.

void insertText (QString text) override

Adds a text.

JsonObject toJson () override

Returns the json representation as JsonObject.

## Private Members

QList<*WebIconTextBannerItem*> \_content

class WebFilePropertyDialogTabTableView : public *sh::ui::FilePropertyDialogTabTableView*, pu

## Public Types

typedef QPair<int, int> ItemIndex

### Public Functions

**WebFilePropertyDialogTabTableView** (*WebFilePropertyDialog \*dlg*)

void **setContent** (QList<QPair<QString, QString>> *content*) **override**  
Sets the content.

QList<*ItemIndex*> **getSelectedItems** () **override**  
Returns the selected cells.

QVariant **getItemValue** (int *r*, int *c*) **override**  
Returns a value of a cell.

QJsonObject **toJson** () **override**  
Returns the json representation as QJsonObject.

void **setSelectedItems** (QList<int> *selitms*)

### Private Members

QList<QPair<QString, QString>> **\_content**

QList<int> **\_selectedItems**

**class WebFilePropertyDialogTabTextView**: public *sh::ui::FilePropertyDialogTabTextView*, public

### Public Functions

**WebFilePropertyDialogTabTextView** (*WebFilePropertyDialog \*dlg*)

void **setContent** (QString *content*) **override**  
Sets the content.

QJsonObject **toJson** () **override**  
Returns the json representation as QJsonObject.

### Private Members

QString **\_content**

**class WebFileView**: public *sh::ui::FileView*  
*#include <webfileview.h>* Web based file view.

### Public Functions

**WebFileView** (*WebFileViewManager \*manager*)

*sh::ui::ColumnDimensions* \***columnDimensions** () **override**  
Returns the column dimensions handler. .

*sh::tools::HistoryTracker*<std::shared\_ptr<const *sh::filesystem::Eurl*>> \***historyTracker** () **override**  
Returns the history tracker. .

*FileViewMode* **viewmode** () **override**  
Returns the view mode (icons, list, ... ?). .



```

void setViewmode (FileViewMode m) override
    Sets the view mode. .

int index () override
    Returns the position of this file view within the panel. .

void gotoDir (std::shared_ptr<sh::filesystem::FilesystemNode> n) override
    Jumps to a new current directory.

    Implementations have to call the base class implementation inside.

sh::filesystem::SizeFormatting getSizeFormattingMode () override
    Returns the mode how file sizes are formatted for displaying. .

void setSizeFormattingMode (sh::filesystem::SizeFormatting mode) override
    Sets the mode how file sizes are formatted for displaying. .

bool hiddenFilesVisible () override
    Returns if hidden files are visible. .

void setHiddenFilesVisible (bool v) override
    Sets the visibility of hidden files. .

int iconDimension () override
    Returns the icon size (in pixels). .

void setIconDimension (int v) override
    Sets the icon size (in pixels). .

void setSort (int column, Qt::SortOrder order) override
    Sets how to sort this view. .

int sortColumn () override
    Returns the index of the current sort column. .

Qt::SortOrder sortOrder () override
    Returns the current sort order. .

QList<std::shared_ptr<sh::filesystem::FilesystemNode>> selectedNodes () override
    Returns the nodes which the user has selected in this fileview. .

void setSelection (const      QList<std::shared_ptr<sh::filesystem::FilesystemNode>>
                    nodes) override
    Sets the node selection of this fileview. .

QList<std::shared_ptr<sh::filesystem::FilesystemNode>> getAllVisibleNodes ()
                                                                    override
    Returns a list of all nodes currently listed in this file view. .

qint64 id ()
    Returns the unique identifier.

std::shared_ptr<sh::filesystem::FilesystemModelFileviewProxy> model ()
    Returns the filesystem model bound to this view.

std::shared_ptr<sh::filesystem::FilesystemNode> node ()
    Returns the current directory.

void reload (bool skipModel = false)
    Reloads the data inside this file view.

sh::filesystem::FilesystemModelFileviewProxy *filemodel ()
    Returns the filesystem model for this view. .

```

```
void setFilemodel (sh::filesystem::FilesystemModelFileviewProxy *model)  
    Sets the filesystem model for this view. .
```

## Signals

```
void currentNodeChanged ()  
void modelChanged ()  
void selectionChanged ()  
    Triggered when the selection have changed.  
void viewOptionChanged ()  
    Triggered when some view options have changed.
```

## Private Functions

```
void _set_model ()  
void setSelectedNode (std::shared_ptr<sh::filesystem::FilesystemNode> node)  
void setCheckedNodes (QList<std::shared_ptr<sh::filesystem::FilesystemNode>> nodes)
```

## Private Members

```
std::shared_ptr<sh::filesystem::FilesystemModelFileviewProxy> _model  
qint64 _id  
FileViewMode _viewmode = FileViewMode::ListView  
sh::filesystem::SizeFormatting _sizeformatting = sh::filesystem::SizeFormatting::SizeFormattingModePrefix  
bool _hiddenFilesVisible = false  
int _iconDimension = 20  
int _sortColumn = 0  
Qt::SortOrder _sortOrder = Qt::SortOrder::AscendingOrder  
std::shared_ptr<sh::filesystem::FilesystemNode> _selectedNode  
QList<std::shared_ptr<sh::filesystem::FilesystemNode>> _checkedNodes  
WebFileViewManager * _manager
```

## Friends

```
friend class WebFileViewManager  
class WebFileViewManager : public QObject, public sh::ui::web::WebModule  
    #include <webfileview.h> Manages WebFileView instances.
```

## Public Functions

bool **executeCommand** (*WebServerEngineRequest* \*request, QString command)  
**override**

Executes command as requested by the browser, if responsible.

Read parameters, check if this module is responsible for answering, and then write an answer with request.

Implementations often use *executeCommand\_byQObjectReflection()* for convenience.

**Return** If it handled (answered) the command.

std::shared\_ptr<*WebFileView*> **addFileView** (bool reinitialize)

void **setCurrentFileViewByIndex** (int idx)

*sh::ui::web::WebFileView* \***currentFileView** ()

int **fileViewCount** ()

void **removeFileView** (int i)

*sh::ui::web::WebFileView* \***getFileView** (int i)

int **getFileViewIndex** (*sh::ui::FileView* \*fileview)

bool **rootCommand** (*WebServerEngineRequest* \*request, QString webadapter)

Returns the root ('index.html') content. .

void **setEngine** (*WebServerEngine* \*webserver)

Assigns this web module to a *sh::ui::web::WebServerEngine*. .

*WebServerEngine* \***webserver** ()

Returns the *sh::ui::web::WebServerEngine* this web module is assigned to (may be nullptr).

## Signals

void **activeSelectionChanged** ()

void **fileViewOptionsChanged** (int i)

void **fileViewCountChanged** ()

void **currentFileViewChanged** ()

## Public Static Functions

bool **executeCommand\_byQObjectReflection** (QObject \*o, *WebServerEngineRequest* \*request, QString command, QString justLeft-side = QString())

Convenience function often used by *executeCommand()*.

For a QObject, it searches there for a slot with the name cmd\_{command}, with ``'s replaced by '\_'s (so, for the command "foo/getBars", it searches for slotcmd\_foo\_getBars). If it finds one, it gets executed (in a worker thread). The request is considered as handled, if so.

If there is such a slot, but with appended \_\_M in name, it is called in main thread!

**Return** If it handled (answered) the command.

### Private Members

```
QList<std::shared_ptr<WebFileView>> _fileviews
int _icurrentfileview = 0
```

### Private Slots

```
void cmd__checkednodes_changed__M(WebServerEngineRequest *request)
void cmd__focus_current__M(WebServerEngineRequest *request)
void cmd__get__M(WebServerEngineRequest *request)
void cmd__list_view_items__M(WebServerEngineRequest *request)
void cmd__node_activated__M(WebServerEngineRequest *request)
void cmd__selection_changed__M(WebServerEngineRequest *request)
```

```
class WebI18n
```

```
    #include <webi18n.h> Web related helpers for internationalization/translations.
```

### Public Static Functions

```
QString get ()
    Returns the Javascript representation for translated strings (in current ui language).
```

```
class WebLogViewDialog : public QObject, public sh::ui::web::WebDialog, public sh::ui::LogViewDialog
    #include <weblogviewdialog.h> Web based log view dialog.
```

### Public Functions

```
WebLogViewDialog (QString headtext, QString subheadtxt, sh::base::LogSeverity min-
                    severity)
```

```
QVariant dialogProperty (QString dlgPropertyKey, QVariant deflt = QVariant())
    Returns a dialog property value by key.
```

```
void setDialogProperty (QString dlgPropertyKey, QVariant value)
    Sets a dialog property value for a key.
```

```
QVariantHash dialogResult ()
    Returns the dialog result as hash.
```

The browser side of a web dialog can ‘accept’ a dialog while closing by setting a non-empty dialog result. This mechanism allows browser side dialogs to return data to the backend. It’s up to the backend to interpret the content of this hash.

If the dialog was cancelled, or is still open, the returned hash is empty. See also wasClosed().

```
void triggerDialogEvent (QString name, QJsonObject data = QJsonObject())
    Triggers a dialog event (at the browsers).
```

They in turn typically fetch some data again in order to refresh the ui.

```
void setCloseableByUser (bool v = true)
    Sets if this dialog shall be directly closeable by the user by the window decoration. Set to
    false if the user has to react on this dialog by clicking on some controls in the dialog body.
    This method must be called during construction.
```

**bool isCloseableByUser ()**  
Returns if this dialog shall be directly closeable by the user by the window decoration.

**QJsonObject toJson () override**  
Returns the json representation as QJsonObject.

**QString headtext ()**  
Returns the header text.

**QString subheadtxt ()**  
Returns the 2nd header text.

**sh::base::LogSeverity minimumSeverity ()**  
Returns the minimum severity.

**qint64 dialogId ()**  
Returns the dialog id.  
  
Each instance has an id unique in the complete Shallot process lifetime.  
  
Must be called in main thread.

**bool isInitiated ()**  
Returns if this dialog is initialized.  
  
Must be called in main thread.

**bool wasAccepted ()**  
Returns if this dialog was 'accepted' by the user (typically by clicking some 'OK' button).  
  
Must be called in main thread.

**void waitClosed ()**  
Wait until the user closed the dialog in some way.  
  
May be called in any thread.

**void close ()**  
Closes the dialog.  
  
Must be called in main thread.

**bool wasClosed ()**  
Returns if this dialog was closed.  
  
Must be called in main thread.

**DialogManager \*manager ()**  
Returns the *DialogManager* which hosts this dialog.

### Private Slots

**void cmd\_\_get\_messages (WebServerEngineRequest \*request)**

**class WebMainWindow : public QObject, public sh::ui::MainWindow, public sh::ui::web::WebModule**  
*#include <webmainwindow.h>* The web main window implementation.

## Public Functions

**WebMainWindow** (QString *dispatcherUrl*, QByteArray *wtoken*)

**~WebMainWindow** ()

*WebServerEngine* \***webserver** ()

Returns the *WebServerEngine* instance for this main window.

void **initialize** () **override**

Initializes this main window. .

std::shared\_ptr<*DialogManager*> **dialogManager** () **override**

Returns the *DialogManager* which creates and drives *Dialogs* for this main window.

You typically should not need to use it directly.

int **fileViewCount** () **override**

Returns the current number of file views. .

void **addFileView** (bool *reinitialize* = false) **override**

Adds a new file view. .

void **removeFileView** (int *i*) **override**

Removes a file view. .

*sh::ui::FileView* \***currentFileView** () **override**

Returns the file view which is currently active (i.e. focussed). .

*sh::ui::FileView* \***getFileView** (int *i*) **override**

Returns a file view by position index. .

void **jumpToNode** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> *node*) **override**

Let the current view jump to another location. .

std::shared\_ptr<*sh::filesystem::FilesystemNode*> **currentDirectory** () **override**

Gets the *sh::filesystem::FilesystemNode* selected in the current view. .

void **setTitlePattern** (QString *value*) **override**

Sets the window title pattern. .

void **setTreeVisible** (bool *v*) **override**

Sets the visibility of the directory tree. .

void **setTreeSticky** (bool *v*) **override**

Sets if the directory tree should follow the active file view. .

void **setFileDetailsPanelVisible** (bool *v*) **override**

Sets the visibility of the file details panel. .

std::shared\_ptr<*sh::ui::ActionExecutionInfoPanel*> **addInfoPanel** (int *position*,  
*sh::actions::ActionExecutionInfo*  
\**info*, QColor *color* =  
QColor()) **override**

Creates a new action execution panel.

Let this smart pointer die for removing it.

void **jumpbarSetTextMode** () **override**

Sets the jumpbar to text input mode. .

void **jumpbarSetButtonMode** () **override**

Sets the jumpbar to button mode. .

```

bool jumpbarIsTextMode () override
    Returns if the jumpbar is in text input mode. .

QString toolbarPosition () override
    Returns the current toolbar position. .

void setToolbarPosition (QString pos) override
    Sets the toolbar position. .

QString detailPanelPosition () override
    Returns the current detail panel position. .

void setDetailPanelPosition (QString pos) override
    Sets the detail panel position. .

std::shared_ptr<AboutDialog> showAboutDialog () override
    Shows and returns an 'About' dialog. .

std::shared_ptr<ManageProfilesDialog> showManageProfilesDialog () override
    Shows and returns a 'Manage Saved Settings' dialog. .

std::shared_ptr<OpenWithDialog> showOpenWithDialog (std::shared_ptr<sh::filesystem::FilesystemNode>
                                                    node,
                                                    QList<std::shared_ptr<sh::tools::filetypes::OpenMe
                                                    openMethods) override

    Shows and returns a 'Open With' dialog. .

std::shared_ptr<StoreProfileDialog> showStoreProfileDialog (std::shared_ptr<const
                                                            sh::filesystem::Eurl>
                                                            eurl) override

    Shows and returns a 'Save Settings' dialog. .

std::shared_ptr<TuningDialog> showTuningDialog () override
    Shows and returns a 'Tuning' dialog. .

std::shared_ptr<LogViewDialog> showLogViewDialog (QString headtext = QString(),
                                                    QString subheadtxt =
                                                    QString(), sh::base::LogSeverity
                                                    minseverity =
                                                    sh::base::LogSeverity::_DEFAULT)
                                                    override

    Shows and returns a 'Log' dialog. .

std::shared_ptr<ManageBookmarksDialog> showManageBookmarksDialog ()
                                                    override

    Shows and returns a 'Manage Bookmarks' dialog. .

std::shared_ptr<FilePropertyDialog> showFilePropertyDialog (QList<std::shared_ptr<sh::filesystem::File
                                                            nodes) override

    Shows and returns a 'File Properties' dialog. .

std::shared_ptr<ActionExecutionInfoDialog> createActionExecutionInfoDialog (sh::actions::ActionEx
                                                                                *info)
                                                                                override

    Creates an action execution dialog. .

void handleCloseAppRejected (QString rejectmsg) override
    React on a rejected application close request (with some feedback message). .

void _initialize (sh::base::SingletonInitializer *singletonInitializer)
    Initializes the main window. For custom initialization logic, see MainWindow.initialize. .

void fileViewsReloadAll ()
    Reload all content in each file view.

```

**void onFileViewOptionsChanged** (std::function<void> int  
> *fcn*, QObject \**owner* = 0) Sets a handler for some view options in a file view changed (optionally bound to an owner lifetime).

**void onCurrentFileViewChanged** (std::function<void>  
> *fcn*) QObject \**owner* = 0) Sets a handler for the currently active file view changed (optionally bound to an owner lifetime).

**void onFileViewCountChanged** (std::function<void>  
> *fcn*) QObject \**owner* = 0) Sets a handler for the number of file views changed (optionally bound to an owner lifetime).

**void onCurrentDirectoryChanged** (std::function<void>  
> *fcn*) QObject \**owner* = 0) Sets a handler for the current directory in the active file view changed (optionally bound to an owner lifetime).

**void onGlobalViewOptionsChanged** (std::function<void>  
> *fcn*) QObject \**owner* = 0) Sets a handler for some global view options changed (optionally bound to an owner lifetime).

**void onCurrentProfileChanged** (std::function<void>  
> *fcn*) QObject \**owner* = 0) Sets a handler for the current profile changed (optionally bound to an owner lifetime).

**void jumpToEurl** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*)  
Let the current view jump to another location. .

**QString titlePattern** ()  
Returns the window title pattern.

**void setCurrentProfile** (QString *profile*)  
Sets the current profile. .

**QString currentProfile** ()  
Returns the current profile.

**void reloadProfile** ()  
Reloads the profile data.

**bool treeVisible** ()  
Returns the visibility of the directory tree.

**bool treeSticky** ()  
Returns if the directory tree follows the active file view.

**bool fileDetailsPanelVisible** ()  
Returns the visibility of the file details panel.

**std::shared\_ptr<sh::ui::ActionExecutionInfoPanel> addInfoPanel** (*sh::actions::ActionExecutionInfo* \**info*, QColor *color* = QColor())  
Creates a new action execution panel.  
Let this smart pointer die for removing it.

**std::shared\_ptr<ActionExecutionInfoPanel> showErrorPanel** ()  
Shows an error panel.

**void \_closeDirectly** ()  
Directly begin application shutdown without checks.

**bool closeApp** ()  
Check if the application may shut down now, and if so, initiate it.



bool **isCloseable** (QString \*msg = nullptr)  
 Check if the application may shut down now.

void **handleClosed** ()  
 Do some cleanup steps just when closing starts. Implementations must call base implementation!

bool **isClosing** ()  
 Returns if the application is currently closing.

bool **rootCommand** (WebServerEngineRequest \*request, QString webadapter)  
 Returns the root ('index.html') content. .

void **setEngine** (WebServerEngine \*webserver)  
 Assigns this web module to a *sh::ui::web::WebServerEngine*. .

## Public Static Functions

bool **tryCreateMainWindow** (MainWindow \*&mainWindow)  
*WebMainWindow* \*mainWindow ()

void **openUrlOnDesktop** (QUrl url)  
 Opens a url on the user desktop, typically in the browser.  
 Note: It obeys the 'openbrowser' ui parameter.

void **setMainWindow** (MainWindow \*mainWindow)  
 Sets the main window instance. .

bool **isReady** ()  
 Returns if this process ui is ready (i.e. is created or runs headless).

bool **runsHeadless** ()  
 Returns if this process runs headless, i.e. without any ui, just for a background process.

void **\_closeDirectly** (*sh::base::SingletonInitializer* \*singletonInitializer)

QString **uiMode** ()  
 Returns the UI mode (e.g. qt, web).

bool **executeCommand\_byQObjectReflection** (QObject \*o, *WebServerEngineRequest* \*request, QString command, QString justLeftSide = QString())  
 Convenience function often used by *executeCommand()*.  
 For a QObject, it searches there for a slot with the name cmd\_{command}, with '/'s replaced by '\_'s (so, for the command "foo/getBars", it searches for slotcmd\_foo\_getBars). If it finds one, it gets executed (in a worker thread). The request is considered as handled, if so.  
 If there is such a slot, but with appended \_\_M in name, it is called in main thread!  
**Return** If it handled (answered) the command.

### Private Functions

```
std::shared_ptr<sh::ui::web::WebActionManager> actionManager ()  
void setCloseable (bool closeable, QString message = QString()) override  
    Sets the closeable state and message.  
void setCurrentEurlByNode (std::shared_ptr<sh::filesystem::FilesystemNode> node)
```

### Private Members

```
bool _treeSticky = true  
bool _detailsPanelVisible = true  
QString _currentProfile  
WebServerEngine * _webserver  
std::shared_ptr<sh::filesystem::FilesystemNode> _currentDirectory = 0  
std::shared_ptr<WebActionManager> _actionManager  
std::shared_ptr<WebDetailPanelManager> _detailPanelManager  
std::shared_ptr<WebDialogManager> _dialogManager  
std::shared_ptr<WebFileViewManager> _fileViewManager  
QString _lastCloseableState
```

### Private Slots

```
void cmd_filesystem_get_icon (WebServerEngineRequest *request)  
void cmd_filesystem_list_items (WebServerEngineRequest *request)  
void cmd_log_as_text (WebServerEngineRequest *request)  
void cmd_log_log_message (WebServerEngineRequest *request)  
void cmd_mainwindow_set_current_directory (WebServerEngineRequest *request)  
void cmd_mainwindow_shallot_icon (WebServerEngineRequest *request)  
void cmd_mainwindow_show_logview (WebServerEngineRequest *request)
```

### Private Static Attributes

```
WebMainWindow * _webMainWindow = nullptr
```

## Friends

```
friend class WebFileViewManager
friend class WebActionExecutionInfoPanel
friend class WebActionExecutionInfoDialog
friend class WebDetailPanelManager
```

```
class WebManageBookmarksDialog : public QObject, public sh::ui::web::WebDialog, public sh::ui::M
#include <webmanagebookmarksdialog.h> Web based manage bookmarks dialog.
```

## Public Functions

```
WebManageBookmarksDialog ()
```

```
QVariant dialogProperty (QString dlgPropertyKey, QVariant deflt = QVariant())
Returns a dialog property value by key.
```

```
void setDialogProperty (QString dlgPropertyKey, QVariant value)
Sets a dialog property value for a key.
```

```
QVariantHash dialogResult ()
Returns the dialog result as hash.
```

The browser side of a web dialog can ‘accept’ a dialog while closing by setting a non-empty dialog result. This mechanism allows browser side dialogs to return data to the backend. It’s up to the backend to interpret the content of this hash.

If the dialog was cancelled, or is still open, the returned hash is empty. See also wasClosed().

```
void triggerDialogEvent (QString name, QJsonObject data = QJsonObject())
Triggers a dialog event (at the browsers).
```

They in turn typically fetch some data again in order to refresh the ui.

```
void setCloseableByUser (bool v = true)
Sets if this dialog shall be directly closeable by the user by the window decoration. Set to
false if the user has to react on this dialog by clicking on some controls in the dialog body.
```

This method must be called during construction.

```
bool isCloseableByUser ()
Returns if this dialog shall be directly closeable by the user by the window decoration.
```

```
QJsonObject toJson () override
Returns the json representation as QJsonObject.
```

```
qint64 dialogId ()
Returns the dialog id.
```

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

```
bool isInitiated ()
Returns if this dialog is initialized.
```

Must be called in main thread.

```
bool wasAccepted ()
Returns if this dialog was ‘accepted’ by the user (typically by clicking some ‘OK’ button).
```

Must be called in main thread.

void **waitClosed** ()

Wait until the user closed the dialog in some way.

May be called in any thread.

void **close** ()

Closes the dialog.

Must be called in main thread.

bool **wasClosed** ()

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \***manager** ()

Returns the *DialogManager* which hosts this dialog.

### Private Slots

void **cmd\_\_add\_bookmark\_\_M** (*WebServerEngineRequest* \*request)

void **cmd\_\_delete\_bookmark** (*WebServerEngineRequest* \*request)

void **cmd\_\_get** (*WebServerEngineRequest* \*request)

void **cmd\_\_move\_bookmark\_down** (*WebServerEngineRequest* \*request)

void **cmd\_\_move\_bookmark\_to\_folder** (*WebServerEngineRequest* \*request)

void **cmd\_\_move\_bookmark\_up** (*WebServerEngineRequest* \*request)

void **cmd\_\_store** (*WebServerEngineRequest* \*request)

```
class WebManageProfilesDialog : public QObject, public sh::ui::web::WebDialog, public sh::ui::Ma
#include <webmanageprofilesdialog.h> Web based manage saved settings dialog.
```

### Public Functions

**WebManageProfilesDialog** ()

QVariant **dialogProperty** (QString *dlgPropertyKey*, QVariant *deflt* = QVariant())

Returns a dialog property value by key.

void **setDialogProperty** (QString *dlgPropertyKey*, QVariant *value*)

Sets a dialog property value for a key.

QVariantHash **dialogResult** ()

Returns the dialog result as hash.

The browser side of a web dialog can ‘accept’ a dialog while closing by setting a non-empty dialog result. This mechanism allows browser side dialogs to return data to the backend. It’s up to the backend to interpret the content of this hash.

If the dialog was cancelled, or is still open, the returned hash is empty. See also wasClosed().

void **triggerDialogEvent** (QString *name*, QJsonObject *data* = QJsonObject())

Triggers a dialog event (at the browsers).

They in turn typically fetch some data again in order to refresh the ui.

void **setCloseableByUser** (bool v = true)

Sets if this dialog shall be directly closeable by the user by the window decoration. Set to `false` if the user has to react on this dialog by clicking on some controls in the dialog body.

This method must be called during construction.

bool **isCloseableByUser** ()

Returns if this dialog shall be directly closeable by the user by the window decoration.

QJsonObject **toJson** () **override**

Returns the json representation as QJsonObject.

qint64 **dialogId** ()

Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

bool **isInitiated** ()

Returns if this dialog is initialized.

Must be called in main thread.

bool **wasAccepted** ()

Returns if this dialog was 'accepted' by the user (typically by clicking some 'OK' button).

Must be called in main thread.

void **waitClosed** ()

Wait until the user closed the dialog in some way.

May be called in any thread.

void **close** ()

Closes the dialog.

Must be called in main thread.

bool **wasClosed** ()

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \***manager** ()

Returns the *DialogManager* which hosts this dialog.

## Private Slots

void **cmd\_get\_M** (*WebServerEngineRequest* \*request)

void **cmd\_icon** (*WebServerEngineRequest* \*request)

void **cmd\_remove\_node\_M** (*WebServerEngineRequest* \*request)

void **cmd\_remove\_profile\_M** (*WebServerEngineRequest* \*request)

**class WebModule**

*#include <webmodule.h>* Base class for modules, which provide some functionality on top of a web server engine.

Subclassed by *sh::ui::web::WebActionManager*, *sh::ui::web::WebDetailPanelManager*,  
*sh::ui::web::WebDialogManager*, *sh::ui::web::WebFileViewManager*,

```
sh::ui::web::WebMainWindow,  
sh::ui::web::WebServerEngineMainModule
```

```
sh::ui::web::WebServerEngineDispatcher,
```

## Public Functions

```
WebModule ()
```

```
~WebModule ()
```

```
bool executeCommand (WebServerEngineRequest *request, QString command)
```

Executes command as requested by the browser, if responsible.

Read parameters, check if this module is responsible for answering, and then write an answer with request.

Implementations often use *executeCommand\_byQObjectReflection()* for convenience.

**Return** If it handled (answered) the command.

```
bool rootCommand (WebServerEngineRequest *request, QString webadapter)
```

Returns the root ('index.html') content. .

```
void setEngine (WebServerEngine *webserver)
```

Assigns this web module to a *sh::ui::web::WebServerEngine*. .

```
WebServerEngine *webserver ()
```

Returns the *sh::ui::web::WebServerEngine* this web module is assigned to (may be nullptr).

## Public Static Functions

```
bool executeCommand_byQObjectReflection (QObject *o, Web-  
ServerEngineRequest *request,  
QString command, QString justLeft-  
side = QString())
```

Convenience function often used by *executeCommand()*.

For a QObject, it searches there for a slot with the name cmd\_{command}, with ``'s replaced by '\_'s (so, for the command"foo/getBars", it searches for slotcmd\_foo\_getBars). If it finds one, it gets executed (in a worker thread). The request is considered as handled, if so.

If there is such a slot, but with appended \_\_M in name, it is called in main thread!

**Return** If it handled (answered) the command.

## Private Members

```
WebServerEngine *_webserver = nullptr
```

```
class WebOpenWithDialog : public QObject, public sh::ui::web::WebDialog, public sh::ui::OpenWithD  
#include <webopenwithdialog.h> Web based open with dialog.
```

## Public Functions

**WebOpenWithDialog** (std::shared\_ptr<sh::filesystem::FilesystemNode> *node*,  
QList<std::shared\_ptr<sh::tools::filetypes::OpenMethod>> *open-  
Methods*)

std::shared\_ptr<sh::tools::filetypes::OpenMethod> **chosenMethod** () **override**  
Returns the chosen open-method (program for opening the file).

bool **rememberForMimetype** () **override**  
Returns if the chosen method is checked to be remembered for the same mime-type in the future.

std::shared\_ptr<const sh::filesystem::Eurl> **rememberForDirectory** () **override**  
If it's checked for the chosen method to be remembered for some subdirectory in the future, it returns the Eurl of this subdirectory, otherwise nullptr.

bool **rememberForFile** () **override**  
Returns if the chosen method is checked to be remembered for the same file in the future.

QVariant **dialogProperty** (QString *dlgPropertyKey*, QVariant *deflt* = QVariant())  
Returns a dialog property value by key.

void **setDialogProperty** (QString *dlgPropertyKey*, QVariant *value*)  
Sets a dialog property value for a key.

QVariantHash **dialogResult** ()  
Returns the dialog result as hash.

The browser side of a web dialog can 'accept' a dialog while closing by setting a non-empty dialog result. This mechanism allows browser side dialogs to return data to the backend. It's up to the backend to interpret the content of this hash.

If the dialog was cancelled, or is still open, the returned hash is empty. See also wasClosed().

void **triggerDialogEvent** (QString *name*, QJsonObject *data* = QJsonObject())  
Triggers a dialog event (at the browsers).

They in turn typically fetch some data again in order to refresh the ui.

void **setCloseableByUser** (bool *v* = true)  
Sets if this dialog shall be directly closeable by the user by the window decoration. Set to false if the user has to react on this dialog by clicking on some controls in the dialog body.  
This method must be called during construction.

bool **isCloseableByUser** ()  
Returns if this dialog shall be directly closeable by the user by the window decoration.

QJsonObject **toJson** () **override**  
Returns the json representation as QJsonObject.

std::shared\_ptr<sh::filesystem::FilesystemNode> **node** ()  
Returns the file node which is later to be opened.

QList<std::shared\_ptr<tools::filetypes::OpenMethod>> **openMethods** ()  
Returns the open-methods (programs for opening the file) to present for choice.

qint64 **dialogId** ()  
Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

bool **isInit** ()

Returns if this dialog is initialized.

Must be called in main thread.

bool **wasAccepted** ()

Returns if this dialog was 'accepted' by the user (typically by clicking some 'OK' button).

Must be called in main thread.

void **waitClosed** ()

Wait until the user closed the dialog in some way.

May be called in any thread.

void **close** ()

Closes the dialog.

Must be called in main thread.

bool **wasClosed** ()

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \***manager** ()

Returns the *DialogManager* which hosts this dialog.

## Private Slots

void **cmd\_\_check\_another\_ancestor\_\_M** (*WebServerEngineRequest* \*request)

void **cmd\_\_check\_custom\_opener\_\_M** (*WebServerEngineRequest* \*request)

void **cmd\_\_open\_method\_icon\_\_M** (*WebServerEngineRequest* \*request)

void **cmd\_\_open\_methods\_\_M** (*WebServerEngineRequest* \*request)

**class WebServerEngine**

*#include <webserverengine.h>* Abstract base class for a web server engine.

Such an engine implements an http server which a browser can connect to.

## Public Functions

**WebServerEngine** (bool *withMainModule*, int *minport*, int *maxport*, QString *externalUrl*,  
QString *dispatcherUrl*, QByteArray *wtoken*)

**~WebServerEngine** ()

QUrl **url** () = 0

Returns the home url of this engine (where the end user can point the browser to). .

QUrl **externalUrl** ()

Returns the external root url for creating full externally valid urls (even behind reverse proxies).

Returns *url()* if no external root url is specified.

Change this by the cmdline parameter '*externalUrl*'.

QUrl **dispatcherUrl** ()

Returns the internal root url of the dispatcher (workers only).



QByteArray **dispatcherWtoken** ()

Returns the wtoken got from the dispatcher (workers only).

QUrl **getFullCommandUrl** (QString *command*, *WebCommandData* *data*, bool *external*)

Generates a url path for a command.

Typically only needed for full url generation on server side.

void **triggerEvent** (QString *name*, QJsonObject *data*)

Triggers an event (at the clients).

void **triggerEvent** (QString *name*, QJsonValue *data* = QJsonValue::Undefined)

QString **getConfigValue** (QString *key*)

Returns the value for a runtime configuration key.

void **setConfigValue** (QString *key*, QString *value*)

Sets the runtime configuration key to a value (triggering a event updating the clients).

void **getStaticFile** (*WebServerEngineRequest* \**request*, QString *path*)

Returns a static file content (for the app itself) on a request.

QUrl **rebaseUrl** (QUrl *rootUrl*, QUrl *url*)

Returns a url rebased to another engine's root url.

Used for dispatching to workers.

void **addAndPreserveModule** (std::shared\_ptr<*sh::ui::web::WebModule*> *module*)

Plugs a *WebModule* into the engine and holds a pointer to it. Unplug it by calling *removeModule()*.

void **addModule** (std::shared\_ptr<*sh::ui::web::WebModule*> *module*)

Plugs a *WebModule* into the engine. Unplug it by either delete module (drop all pointers to it) or by calling *removeModule()*.

void **removeModule** (*sh::ui::web::WebModule* \**module*)

Unplugs a *WebModule* from the engine.

qint64 **currentTimestamp** ()

Returns the current engine timestamp.

It's not related to real time, but just a value which is higher each time you query it.

This is used for coordination of some time-order sensitive operations on browser side.

## Public Static Functions

*WebServerEngine* \***createDispatcherEngine** ()

Creates and returns a web engine for a dispatcher.

*WebServerEngine* \***createWorkerEngine** (QString *dispatcherUrl*, QByteArray *wtoken*)

Creates and returns a web engine for a worker.

### Parameters

- *dispatcherUrl*: The root url of the dispatcher.

### Private Members

```
const QJsonObject _jok
QMutex _configValuesMutex
QMutex _modulesMutex
int _minport
int _maxport
QUrl _externalurl
QUrl _dispatcherurl
QMap<QString, QString> _configValues
QList<std::weak_ptr<sh::ui::web::WebModule>> _modules
QList<std::shared_ptr<sh::ui::web::WebModule>> _smodules
std::shared_ptr<sh::ui::web::WebServerEngineMainModule> _mainmodule
QString _webstaticpath
qint64 _currentTimestamp = 1
QByteArray _wtoken
```

### Private Static Functions

```
WebServerEngine *_createEngine (bool withMainModule, QString dispatcherUrl,
                                QByteArray wtoken)
```

### Friends

```
friend class WebServerEngineMainModule
```

```
class WebServerEngineDispatcher : public QObject, public sh::ui::web::WebModule
#include <webserverenginedispatcher.h> The web serve engine dispatcher module, used in dispatcher mode for providing a portal url which starts Shallot instances on request and internally redirects to them.
```

### Public Functions

```
WebServerEngineDispatcher ()
```

```
~WebServerEngineDispatcher ()
```

```
void stop ()
    Stops the dispatcher.
```

```
void setEngine (WebServerEngine *webserver)
    Assigns this web module to a sh::ui::web::WebServerEngine. .
```

```
WebServerEngine *webserver ()
    Returns the sh::ui::web::WebServerEngine this web module is assigned to (may be nullptr).
```

## Public Static Functions

*WebMainWindow* \***handleDispatching** (std::function<*WebMainWindow*\*> QString dispatcherUrl, QByteArray wtoken  
 > *createWndFctHandles* dispatching and creates a new main window (in child processes) on demand via a given factory function.

void **doInitialize** ()

void **doShutdown** ()

bool **executeCommand\_byQObjectReflection** (QObject \*o, *WebServerEngineRequest* \*request, QString command, QString justLeftSide = QString())

Convenience function often used by *executeCommand()*.

For a QObject, it searches there for a slot with the name cmd\_{command}, with ``'`'s replaced by '\_'s (so, for the command "foo/getBars", it searches for slotcmd\_foo\_getBars). If it finds one, it gets executed (in a worker thread). The request is considered as handled, if so.

If there is such a slot, but with appended \_\_M in name, it is called in main thread!

**Return** If it handled (answered) the command.

## Private Functions

bool **compareHashRedirectorWebworkerUrl** (QByteArray hash, QString url, QString wtoken)

Compares a hash for a redirector webworker url to other data.

*WorkerProcess* \***spawnNewWorker** (*WebServerEngineRequest* \*request)

Spawns a new worker.

Does not execute any requests on it.

*WorkerProcess* \***findWorkerForWtoken** (QString wtoken)

Returns the worker for a wtoken.

QString **wtokenFromRequestCookie** (*WebServerEngineRequest* \*request)

Returns the wtoken from a request (typically from cookie).

bool **executeCommand** (*WebServerEngineRequest* \*request, QString command) **override**

Executes command as requested by the browser, if responsible.

Read parameters, check if this module is responsible for answering, and then write an answer with request.

Implementations often use *executeCommand\_byQObjectReflection()* for convenience.

**Return** If it handled (answered) the command.

bool **rootCommand** (*WebServerEngineRequest* \*request, QString webadapter) **override**

Returns the root ("index.html") content. .

### Private Members

```
std::shared_ptr<WebServerEngine> _webserver  
QMutex _workersmutex  
QHash<QString, WorkerProcess*> _workers  
JanitorThread _janitor
```

### Private Slots

```
void cmd__drop_worker__M (WebServerEngineRequest *request)  
void cmd__set_last_user_activity (WebServerEngineRequest *request)  
void cmd__set_prevent_close_message (WebServerEngineRequest *request)  
void cmd__working_for (WebServerEngineRequest *request)
```

### Private Static Functions

```
void start (QUrl &url)  
    Starts the dispatcher engine, accepting http connections and internally handling child processes.  
QByteArray hashRedirectorWebworkerUrl (QString url, QString wtoken)  
    Compute a hash for a redirector webworker url.
```

### Private Static Attributes

```
std::shared_ptr<WebServerEngineDispatcher> _dispatcher = nullptr  
QString _dispatcherId = QString::fromLatin1(sh::tools::Misc::generateUniqueHash())  
QString _dispatcherCookieName = QUrl::toPercentEncoding(QString("shallot_%1").arg(_dispatcherId)).toLatin1()  
class JanitorThread: public QThread
```

### Public Functions

```
JanitorThread (WebServerEngineDispatcher *dispatcher)  
void run () override
```

### Private Functions

```
qint64 machineMemory ()  
    Returns the amount of memory this machine provides (in bytes).  
    Returns -1 if this is not supported on the target platform or it failed.  
void stopWorker (WorkerProcess *worker, QString reason)  
int detectGoodMaxNumberWorkersByMachineCapabilities ()  
QList<WorkerProcess*> stopWorkers (QList<WorkerProcess*> workers, int stopcount,  
    QString reason)
```

## Private Members

*WebServerEngineDispatcher* \*\_dispatcher

double \_allMemory

QHash<*WorkerProcess*\*, double> \_workerMemoryUsagesLastSeconds

**class WorkerProcess : public QProcess**

A internal Shallot worker process driving one Shallot session.

## Public Functions

QByteArray **wtoken** ()

The wtoken (used for association and security).

QUrl **rootUrl** ()

The worker root url.

void **request** (*WebServerEngineRequest* \*request)

Make a request to the worker.

qint64 **workerPid** ()

Returns the process id of the worker.

qint64 **memoryUsage** ()

Returns the current memory usage of this worker (in bytes).

Returns -1 if this is not supported on the target platform or it failed.

QDateTime **lastBrowserHeartbeat** ()

Returns when a browser was seen connected to this worker last time.

QDateTime **lastUserInteraction** ()

Returns when a user has interacted with this worker last time.

QString **clientHost** ()

Returns the client host this worker is serving.

It's not guaranteed to have some specific content (maybe an ip address), but is equal for the same host, some one can count how many hosts work for one particular client host.

QString **preventCloseMessage** ()

Returns a reason message why this worker currently should not be closed (or "" if closing is fine).

See also `isClosingInappropriate()`.

Note: This is reported asynchronously. You must not rely on its precise information, but solely use it for monitoring.

bool **isClosingFine** ()

Returns if it is fine to stop this worker (or if there is a good reason to not do).

See also `preventCloseMessage()`.

Note: This is reported asynchronously. You must not rely on its precise information, but solely use it for monitoring.

## Private Functions

**WorkerProcess** (*WebServerEngineDispatcher* \*dispatcher, QByteArray wtoken, QString clientHost, QStringList acceptedLanguages)

bool **waitOnline** ()

Waits until the worker is online and ready for requests (or dead).

void **initialize** (QString rooturl, qint64 pid)

Initializes the worker instance (which exists on the dispatcher side!) by setting some final data.

void **gotBrowserHeartbeat** ()

Refreshes the browser heartbeat timestamp.

See *lastBrowserHeartbeat()*.

void **gotUserInteraction** (int value)

Refreshes the user interaction timestamp.

See *lastUserInteraction()*.

void **setPreventCloseMessage** (QString message)

Sets the preventCloseMessage (empty: closeable).

## Private Members

*WebServerEngineDispatcher* \*\_dispatcher

QByteArray \_wtoken

QUrl \_rooturl

QMutex \_mutex

QWaitCondition \_cnd\_initied

qint64 \_pid = -1

QDateTime \_lastBrowserHeartbeat

QDateTime \_lastUserInteraction

QString \_clientHost

QString \_preventCloseMessage

QRegularExpression \_reProcStatus

## Friends

**friend class** WebServerEngineDispatcher

**class** WebServerEngineMainModule : public QObject, public *sh::ui::web::WebModule*  
*#include <webservengine.h>* The web serve engine main module, providing some base services like events.

## Public Functions

**WebServerEngineMainModule** ()

**~WebServerEngineMainModule** ()

bool **executeCommand** (*WebServerEngineRequest* \*request, QString command) **override**

Executes command as requested by the browser, if responsible.

Read parameters, check if this module is responsible for answering, and then write an answer with request.

Implementations often use *executeCommand\_byQObjectReflection()* for convenience.

**Return** If it handled (answered) the command.

bool **rootCommand** (*WebServerEngineRequest* \*request, QString webadapter) **override**

Returns the root ('index.html') content. .

void **triggerEvent** (QString name, QString data)

Triggers an event on browser side.

void **setEngine** (*WebServerEngine* \*webserver)

Assigns this web module to a *sh::ui::web::WebServerEngine*. .

*WebServerEngine* \*webserver ()

Returns the *sh::ui::web::WebServerEngine* this web module is assigned to (may be nullptr).

## Public Static Functions

bool **executeCommand\_byQObjectReflection** (QObject \*o, *WebServerEngineRequest* \*request, QString command, QString justLeftSide = QString())

Convenience function often used by *executeCommand()*.

For a QObject, it searches there for a slot with the name cmd\_{command}, with ``'`s replaced by'\_s (so, for the command"foo/getBars", it searches for slotcmd\_foo\_getBars`). If it finds one, it gets executed (in a worker thread). The request is considered as handled, if so.

If there is such a slot, but with appended \_\_M in name, it is called in main thread!

**Return** If it handled (answered) the command.

## Private Functions

qint64 **lastRequestTime** ()

### Private Members

qint64 **\_lastRequestTime**  
QMutex **\_mutex\_lastRequestTime**  
QList<*EventEntry*\*> **\_eventEntries**  
qint64 **\_eventEntriesLastId** = 0  
QMutex **\_mutex\_eventEntries**  
QWaitCondition **\_eventEntriesCondition**  
*EventEntryCleanupThread* **\_eventEntryCleanupThread**  
*StopWhenIdleThread* **\_stopWhenIdleThread**

### Private Slots

void **cmd\_dispatcher\_toclient\_heartbeat** (*WebServerEngineRequest* \*request)  
void **cmd\_events\_get\_next** (*WebServerEngineRequest* \*request)

### Private Static Attributes

QRegularExpression **\_restatic**  
**class EventEntry**  
Data for one occurred event.

### Public Functions

**EventEntry** (qint64 *id*, qint64 *time*, QString *name*, QString *data*)

### Public Members

**const** qint64 **id**  
**const** qint64 **time**  
**const** QString **name**  
**const** QString **data**

### Friends

**friend class** EventEntryCleanupThread  
**class EventEntryCleanupThread**: **public** QThread  
Thread for cleaning up old *EventEntry* instances.



## Public Functions

**EventEntryCleanupThread** (*WebServerEngineMainModule* \*module)

## Private Members

*WebServerEngineMainModule* \*module

**class StopWhenIdleThread**: public QThread  
Thread for stopping the application if the remote side is gone.

## Public Functions

**StopWhenIdleThread** (*WebServerEngineMainModule* \*module)

## Private Members

*WebServerEngineMainModule* \*module

**class WebServerEngineRequest**  
*#include <webserverengine.h>* A web request in a *WebServerEngine*.  
Override this together with *WebServerEngine*.

## Public Functions

**WebServerEngineRequest** (QString *url*, QString *clientHost*)

QUrl **url** ()  
Returns the requested url.

Note: The url should be considered as opaque, since the exact structure depends on the engine implementation.

QString **clientHost** ()  
Returns the client host address.

QString **path** ()  
Returns the path part of the requested url.

Note: The url should be considered as opaque, since the exact structure depends on the engine implementation.

QString **data** (QString *key*)  
Returns data passed as parameter along with the request by *key*.

int **responseCode** ()  
Returns the answered (http) response code.

QString **responseMimeType** ()  
Returns the mimetype of the answer content.

QByteArray **response** ()  
Returns the answer content.

bool **responseSet** ()  
Returns if a response was set.

```
void setResponse (QByteArray response, QString mimetype, int responseCode = HTTP_OK)
```

Sets the response.

```
void setResponse (QJsonArray response, int responseCode = HTTP_OK)
```

```
void setResponse (QJsonObject response, int responseCode = HTTP_OK)
```

```
QString headerData (QString key) = 0
```

Returns an http header value by key. .

```
QString getCookie (QString key) = 0
```

Returns an http cookie stored on browser side.

Note: This only works on dispatcher level and not in usual *WebModule* commands.

```
void setCookie (QString key, QString value, int maxAgeSecs = -1) = 0
```

Sets an http cookie to be returned to the browser side.

Note: This only works on dispatcher level and not in usual *WebModule* commands.

### Public Static Attributes

```
const int HTTP_OK = 200
```

```
const int HTTP_SEE_OTHER = 303
```

```
const int HTTP_NOT_FOUND = 404
```

```
const int HTTP_INTERNAL_SERVER_ERROR = 500
```

```
const int HTTP_BAD_GATEWAY = 502
```

### Private Members

```
QUrl _url
```

```
QString _clientHost
```

```
QString _path
```

```
QMap<QString, QString> _getdata
```

```
int _responseCode = HTTP_NOT_FOUND
```

```
QString _responseMimeType = "application/octet-stream"
```

```
QByteArray _response
```

```
bool _responseSet = false
```

```
class WebStoreProfileDialog : public QObject, public sh::ui::web::WebDialog, public sh::ui::StoreF  
    #include <webstoreprofiledialog.h> Web based save settings dialog.
```

## Public Functions

**WebStoreProfileDialog** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*)

QList<SettingEntry> **checkedSettings** () **override**

Returns the list of settings the user has checked to store.

QString **profileName** () **override**

Returns the profile the user has chosen to store settings for.

bool **withSubDirectories** () **override**

Returns if the user has chosen to apply the checked settings also for subdirectories.

QStringList **inheritsFrom** () **override**

Returns the list of profiles the user has chosen to inherit from.

bool **hasGlobal** () **override**

Returns if the user has chosen to apply the checked settings for each directory (instead of the current directory).

QVariant **dialogProperty** (QString *dlgPropertyKey*, QVariant *deflt* = QVariant())

Returns a dialog property value by key.

void **setDialogProperty** (QString *dlgPropertyKey*, QVariant *value*)

Sets a dialog property value for a key.

QVariantHash **dialogResult** ()

Returns the dialog result as hash.

The browser side of a web dialog can ‘accept’ a dialog while closing by setting a non-empty dialog result. This mechanism allows browser side dialogs to return data to the backend. It’s up to the backend to interpret the content of this hash.

If the dialog was cancelled, or is still open, the returned hash is empty. See also wasClosed().

void **triggerDialogEvent** (QString *name*, QJsonObject *data* = QJsonObject())

Triggers a dialog event (at the browsers).

They in turn typically fetch some data again in order to refresh the ui.

void **setCloseableByUser** (bool *v* = true)

Sets if this dialog shall be directly closeable by the user by the window decoration. Set to *false* if the user has to react on this dialog by clicking on some controls in the dialog body.

This method must be called during construction.

bool **isCloseableByUser** ()

Returns if this dialog shall be directly closeable by the user by the window decoration.

QJsonObject **toJson** () **override**

Returns the json representation as QJsonObject.

std::shared\_ptr<const *sh::filesystem::Eurl*> **eurl** ()

Returns the current directory this dialog shall work on.

qint64 **dialogId** ()

Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

bool **isInitied** ()

Returns if this dialog is initialized.

Must be called in main thread.

bool **wasAccepted** ()

Returns if this dialog was ‘accepted’ by the user (typically by clicking some ‘OK’ button).

Must be called in main thread.

void **waitClosed** ()

Wait until the user closed the dialog in some way.

May be called in any thread.

void **close** ()

Closes the dialog.

Must be called in main thread.

bool **wasClosed** ()

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \***manager** ()

Returns the *DialogManager* which hosts this dialog.

### Private Slots

void **cmd\_\_get\_\_M** (*WebServerEngineRequest* \*request)

void **cmd\_\_store\_mode\_\_M** (*WebServerEngineRequest* \*request)

**class WebTuningDialog**: public QObject, public *sh::ui::web::WebDialog*, public *sh::ui::TuningDialog*  
*#include <webtuningdialog.h>* Web based tuning dialog.

### Public Functions

**WebTuningDialog** ()

QVariant **dialogProperty** (QString *dlgPropertyKey*, QVariant *deflt* = QVariant())

Returns a dialog property value by key.

void **setDialogProperty** (QString *dlgPropertyKey*, QVariant *value*)

Sets a dialog property value for a key.

QVariantHash **dialogResult** ()

Returns the dialog result as hash.

The browser side of a web dialog can ‘accept’ a dialog while closing by setting a non-empty dialog result. This mechanism allows browser side dialogs to return data to the backend. It’s up to the backend to interpret the content of this hash.

If the dialog was cancelled, or is still open, the returned hash is empty. See also wasClosed().

void **triggerDialogEvent** (QString *name*, QJsonObject *data* = QJsonObject())

Triggers a dialog event (at the browsers).

They in turn typically fetch some data again in order to refresh the ui.

void **setCloseableByUser** (bool *v* = true)

Sets if this dialog shall be directly closeable by the user by the window decoration. Set to *false* if the user has to react on this dialog by clicking on some controls in the dialog body.

This method must be called during construction.

bool **isCloseableByUser** ()

Returns if this dialog shall be directly closeable by the user by the window decoration.

QJsonObject **toJson** () **override**

Returns the json representation as QJsonObject.

qint64 **dialogId** ()

Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

bool **isInitd** ()

Returns if this dialog is initialized.

Must be called in main thread.

bool **wasAccepted** ()

Returns if this dialog was 'accepted' by the user (typically by clicking some 'OK' button).

Must be called in main thread.

void **waitClosed** ()

Wait until the user closed the dialog in some way.

May be called in any thread.

void **close** ()

Closes the dialog.

Must be called in main thread.

bool **wasClosed** ()

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \***manager** ()

Returns the *DialogManager* which hosts this dialog.

## Private Members

QMutex **\_cfgvaluesmutex**

QWaitCondition **\_cfgvaluescondition**

bool **\_cfgvaluesinitd** = false

QList<std::shared\_ptr<*sh::configuration::ConfigurationValue*>> **\_cfgvalues**

## Private Slots

void **cmd\_\_change\_configurationvalue** (*WebServerEngineRequest* \*request)

void **cmd\_\_get\_configurationvalues** (*WebServerEngineRequest* \*request)

### Private Static Functions

QString **categoryToName** (*sh::configuration::ConfigurationCategory* category)

QString **valueTypeToName** (*sh::configuration::ConfigurationValueType* \*valueType)

QJsonObject **configurationValueToJsonObject** (std::shared\_ptr<*sh::configuration::ConfigurationValue*> cfgval)

### namespace engines

Web server low-level engine implementations.

## 10.2.3 Namespace sh::actions

### namespace sh::actions

Infrastructure for actions (submenus, executable ones, and more), as shown in the toolbar and menus and used at some other places.

See the abstract base class *sh::actions::AbstractActionItem* for more.

### class AbstractActionFactory

*#include <actionfactory.h>* Abstract base class for an action factory.

It creates an instance of a certain subclass of *AbstractActionItem*. It can also check if it is defined to be created in a given situation (e.g. not all actions are visible in toolbar).

Subclassed by *sh::actions::ActionFactory< T >*, *sh::scripting::api::ApiActionFactory*

### Public Functions

**AbstractActionFactory** (std::shared\_ptr<*sh::actions::ActionCategory*> category,  
QList<std::shared\_ptr<*Predicate*>> predicates)

Is (for subclasses) intended to be directly constructed and registered once.

**~AbstractActionFactory** ()

std::shared\_ptr<*ActionInstantiation*> **actionAvailable** (*ActionInstantiation* \*instantiation)

std::shared\_ptr<*sh::actions::AbstractActionItem*> **construct** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> nodes) = 0

### Private Members

std::shared\_ptr<*sh::actions::ActionCategory*> **category**

QList<std::shared\_ptr<*Predicate*>> **predicates**

**class AbstractActionItem**: public QObject, public std::enable\_shared\_from\_this<*AbstractActionItem*>

*#include <abstractactionitem.h>* Abstract base class for executable actions, submenus of them, and more.

See the subclasses of this class.

They are used at various places. The toolbar and context menus provide them to the user.

They can be registered e.g. in a *sh::actions::ActionFactory* or returned from *sh::filesystem::FilesystemHandler.getActions* in order to make them available.

Subclassed by *sh::actions::ActionActionItem*, *sh::actions::HeaderActionItem*,  
*sh::actions::SeparatorActionItem*, *sh::actions::SubmenuActionItem*

## Public Functions

QString **text** ()

Returns the displayed text for this action.

QString **icon** ()

Returns the icon for this action.

bool **enabled** ()

Checks if this action is enabled.

bool **isChecked** ()

Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>

> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

**AbstractActionItem** ()

## Signals

void **changed** ()  
Emits when some data changed.

## Private Members

QString **\_text**  
QString **\_icon**  
bool **\_enabled**  
bool **\_visible**  
bool **\_ischeckable**  
bool **\_ischecked**  
bool **\_initialized**  
bool **\_initializing**  
bool **\_reinitialize**  
std::weak\_ptr<AbstractActionItem> **\_parentAction** = NO\_PARENT  
int **\_defaultActionPrecedence**

## Private Static Attributes

QWaitCondition **initcondition**

**class ActionActionItem**: public *sh::actions::AbstractActionItem*  
*#include <actionactionitem.h>* Abstract base class for an executable action.

It can be made visible in menus or the toolbar or executed directly.

Subclasses provide the actual implementation by implementing `action`. Execution can be triggered from outside by calling `execute`.

Subclassed by *sh::actions::common::ActionAbstractTransferTree*, *sh::actions::common::ActionAddToBookmarks*,  
*sh::actions::common::ActionBookmark*, *sh::actions::common::ActionClipboardCopy*,  
*sh::actions::common::ActionClipboardCut*, *sh::actions::common::ActionClipboardPaste*,  
*sh::actions::common::ActionClipboardPasteAsLink*, *sh::actions::common::ActionCreateFile*,  
*sh::actions::common::ActionCreateFolder*, *sh::actions::common::ActionDeleteItems*,  
*sh::actions::common::ActionExecute*, *sh::actions::common::ActionManageBookmarks*,  
*sh::actions::common::ActionNavigateInHistory*, *sh::actions::common::ActionOpenArchive*,  
*sh::actions::common::ActionOpenDirectoryInNewWindow*, *sh::actions::common::ActionOpenFile*,  
*sh::actions::common::ActionOpenFileWith::\_ActionOpenFileWithEntry*, *sh::actions::common::ActionOpenFileWith::\_ActionOpenFileWithEntry*,  
*sh::actions::common::ActionOpenSharcArchive*, *sh::actions::common::ActionOpenTerminalAsUser*,  
*sh::actions::common::ActionRenameItem*, *sh::actions::common::ActionShowProperties*,  
*sh::actions::common::ActionTransferToHelper*, *sh::actions::mainmenu::ActionAbout*,  
*sh::actions::mainmenu::ActionAbstractSelectNodes*, *sh::actions::mainmenu::ActionApplyProfile::ActionApplyProfileX*,  
*sh::actions::mainmenu::ActionFileViewChangeIconDimension*, *sh::actions::mainmenu::ActionFileviewFilesizeFormatting*,  
*sh::actions::mainmenu::ActionFileviewShowHiddenFiles*, *sh::actions::mainmenu::ActionFileviewView*,  
*sh::actions::mainmenu::ActionGotoDirectory*, *sh::actions::mainmenu::ActionHelp*,  
*sh::actions::mainmenu::ActionManageSavedSettings*, *sh::actions::mainmenu::ActionNumberFileviews\_Add*,  
*sh::actions::mainmenu::ActionNumberFileviews\_Remove*, *sh::actions::mainmenu::ActionNumberFileviews\_Remove\_Current*



```

sh::actions::mainmenu::ActionQuit,
sh::actions::mainmenu::ActionSaveSettings,
sh::actions::mainmenu::ActionSetWindowTitlePattern,
sh::actions::mainmenu::ActionShowFolderTree,
sh::actions::mainmenu::ActionTreeStickyness,
sh::filepropertydialogtabs::FilePropertyDialogTabExtendedAttributes::ActionAddAttribute,
sh::filepropertydialogtabs::FilePropertyDialogTabExtendedAttributes::ActionChangeAttribute,
sh::filepropertydialogtabs::FilePropertyDialogTabExtendedAttributes::ActionRemoveAttribute,
sh::filepropertydialogtabs::FilePropertyDialogTabUnixPermissions::ActionChange,
sh::filesystemhandlers::ActionMountGnomeIOLocation, sh::filesystemhandlers::ActionMountNetworkGnomeIOLocation,
sh::filesystemhandlers::ActionUnmountGnomeIOLocation, sh::scripting::api::ApiActionActionItem,
sh::scripting::ScriptingEngine::ActionPluginLoadCrashedAgain, sh::search::criteria::ActionAbstractActionDrivenSearchC

```

## Public Functions

**ActionActionItem** (QString *text*, bool *enabled* = true, QString *icon* = QString(), int *defaultActionPrecedence* = 0, bool *checkable* = false, bool *ischecked* = false)

Is (for subclasses) intended to be directly constructed from everywhere or by registering a factory somewhere.

void **execute** ()

Executes this action.

void **execute** (sh::actions::ActionExecutionInfo \**info*)

Executes this action.

QKeySequence **shortcuthint** ()

Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()

Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)

Sets the keyboard shortcut for triggering this action.

QString **text** ()

Returns the displayed text for this action.

QString **icon** ()

Returns the icon for this action.

bool **enabled** ()

Checks if this action is enabled.

bool **isChecked** ()

Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<AbstractActionItem> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<AbstractActionItem> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void> > *oninitialized* = 0)  
Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

### class ActionCategory

*#include <actioncategory.h>* Action categories are used for grouping actions in menus.

They provide the primary grouping in the toolbar and the context menus. Per default, the categories "create", "open" and "manage" exist.

## Public Functions

**ActionCategory** (QString *name*, QString *label*, QString *icon*)  
Constructed only by the infrastructure and made available otherwise.

QString **name** ()  
The category name.

QString **label** ()  
The category label.

QString **icon** ()  
The category icon.

## Public Static Functions

`std::shared_ptr<ActionCategory> getByName (QString name)`  
 Gets a category by name.

`std::shared_ptr<ActionCategory> add (QString name, QString label, QString icon)`  
 Adds a new category.

`QList<std::shared_ptr<ActionCategory>> categories ()`  
 Returns a list of all existing categories.

`void doInitialize ()`

`void doShutdown ()`

## Private Members

QString **\_name**

QString **\_label**

QString **\_icon**

## Private Static Attributes

`QList<std::shared_ptr<ActionCategory>> _categories`

**class ActionDefaultPrecedenceValues**

*#include <abstractactionitem.h>* Reference values for calculating default precedence values of actions.

## Public Static Attributes

`const int PRECEDENCE_OPENFILE = 300000`  
 Normal file open action.

`const int PRECEDENCE_BUILTIN_SPECIAL_OPEN = 600000`  
 Special open action.

**class ActionExecutionInfo : public QObject**

*#include <actionexecutioninfo.h>* Programming interface for giving status infos and communication with the user in an action execution.

An instance is always available during an execution.

## Public Functions

**ActionExecutionInfo** (QObject *\*parent* = 0)  
 Constructed only by the infrastructure and made available otherwise.

**~ActionExecutionInfo** ()

void **setVisualProcessFeedbackActive** (bool *active*)  
 Specifies if there must be a visual process feedback.

The visual process feedback is typically realized as a dialog showing progress information.

Note: It can also appear in some situations, even if this property is set to false.

**bool isVisualProcessFeedbackActive ()**  
Determines if there must be a visual process feedback.

**void setManualInterventionNeeded (bool *isneeded*)**  
Specifies if manual input is currently required.

**bool isManualInterventionNeeded ()**  
Determines if manual input is currently required.

**void startExecution (std::shared\_ptr<*sh::actions::AbstractActionItem*> *action*)**  
Triggers the beginning of the actual execution. .

**void finishExecution ()**  
Triggers the finishing of the execution. .

**void setDetails (QString *fromverb*, QString *fromobj*, QString *toverb* = QString(), QString *toobj* = QString())**  
Sets some detail texts.

**QString fromVerb ()**  
Gets detail text ‘from-verb’.

**QString fromObjectName ()**  
Gets detail text ‘from-objectname’.

**QString toVerb ()**  
Gets detail text ‘to-verb’.

**QString toObjectName ()**  
Gets detail text ‘to-objectname’.

**void setHead (QString *text*)**  
Sets the header text.

**QString head ()**  
Gets the header text.

**void setProgressIndeterminate (QString *text* = QString())**  
Sets progress information to ‘indeterminate’.

**void setProgress (quint64 *done*, quint64 *all*, QString *text*)**  
Sets progress information.

**bool isProgressDeterminate ()**  
Gets if progress information is ‘indeterminate’.

**quint64 progressDone ()**  
Gets count of finished items.

**quint64 progressAll ()**  
Gets count of all items.

**QString progressText ()**  
Gets progress information text.

***sh::actions::ActionExecutionUserFeedback* \*userfeedback ()**  
Program interface for interactive user input.

**bool isCancelled ()**  
Gets if the current action execution was cancelled.

**void cancel ()**  
Cancel the current action execution.  
Used from outside, e.g. as handler for a ‘Cancel’ button.

void **respectCancel** ()

Called regularly from an action execution code in order to cancel execution at that place, if the user has cancelled it.

void **withExecuteGuards\_infodlg** (std::function<void>)

>int *flags* = 0 Executes code with different exception handlers for presenting exceptions in the action execution inside the action dialog (instead of the default exception dialog).

*sh::filesystem::Operation* \***operation** ()

Program interface for operations on the filesystem (reading or writing).

void **addChangedEurl** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*)

Reminds an important change of a certain node in the filesystem (file created, modified, removed?).

The infrastructure will force the ui to refresh those nodes.

void **shutdown** (bool *success*)

Shuts down this execution info and releases resources.

## Signals

void **detailsChanged** ()

Signal for changed execution details.

void **headChanged** ()

Signal for changed head.

void **progressChanged** ()

Signal for changed progress.

void **visualProcessFeedbackActiveChanged** ()

Signal for changed visibility-required.

void **wasCancelled** ()

Signal for cancellation on user behalf.

bool **manualInterventionNeededChanged** ()

Signal for changed manual-intervention-needed.

## Private Functions

std::shared\_ptr<*sh::ui::ActionExecutionInfoDialog*> **createDialog** ()

std::shared\_ptr<*ui::ActionExecutionInfoPanel*> **createPanel** (*sh::ui::ActionExecutionInfoDialog* \**dialog*)

## Private Members

QMutex **mutex**

QString **\_head**

QString **\_fromverb**

QString **\_fromobj**

QString **\_toverb**

QString **\_toobj**

```
bool _progressDeterminate
quint64 _progressDone
quint64 _progressAll
bool _visualProcessFeedbackActive = false
bool _wasAlwaysInvisible = true
bool _manualInterventionNeeded = false
QString _progressText
std::shared_ptr<sh::ui::ActionExecutionInfoDialog> infodialog = 0
std::shared_ptr<sh::ui::ActionExecutionInfoPanel> infopanel = 0
bool _iscancelled = false
sh::filesystem::Operation _operation
QList<std::shared_ptr<const sh::filesystem::Eurl>> changes
QStack<std::shared_ptr<sh::actions::AbstractActionItem>> stack
```

### Private Slots

```
void _checkVisibilityOnce ()
```

#### **class ActionExecutionUserFeedback**

*#include <actionexecutionuserfeedback.h>* Methods for user interaction in an action execution.

When an action executes, it may need to ask the user for some input at some time. An action implementation can do so by calling those methods. It is available as a member of the *ActionExecutionInfo* class. An instance of it is available in each action execution.

Subclassed by *sh::ui::ActionExecutionInfoDialog*

### Public Types

#### **enum MessageBoxButton**

Buttons in a message box from *ActionExecutionUserFeedback*.

*Values:*

```
enumerator NONE = 0
enumerator OK = 1 << 0
enumerator Continue = 1 << 1
enumerator Cancel = 1 << 2
enumerator Retry = 1 << 3
enumerator Yes = 1 << 4
enumerator No = 1 << 5
```

## Public Functions

```

int messageBox (QString text, QList<QString> answers, QString icon = QString(), int defaultanswer = -1, int cancelanswer = -1, QList<QString> answericons = QList<QString>()) = 0

int inputBox (QString text, QList<QString> answers, QString *value, QString icon = QString(), int defaultanswer = -1, int cancelanswer = -1, int valuePreselectFrom = -1, int valuePreselectTo = -1) = 0

int multilineInputBox (QString text, QList<QString> answers, QString *value, QString icon = QString(), int defaultanswer = -1, int cancelanswer = -1) = 0

int simpleChooserGridform (QString text, GridformEntries *entries, QStringList answers, int defaultanswer = -1, int cancelanswer = -1) = 0

bool credentialsDialog (QString text, bool showDomain, bool showUsername, bool showPassword, bool showAnonymous, bool showRemember, QString *domain, QString *username, QString *password, bool *isAnonymous, bool *isRemember) = 0

bool unixPermissionsDialog (bool *userMayRead, bool *userMayWrite, bool *userMayExecute, bool *groupMayRead, bool *groupMayWrite, bool *groupMayExecute, bool *othersMayRead, bool *othersMayWrite, bool *othersMayExecute, bool *sticky, bool *setuid, bool *setgid, QStringList users, QStringList groups, QString *ownerUser, QString *ownerGroup) = 0

QString simpleInputBox (QString text, QString deft, int valuePreselectFrom = -1, int valuePreselectTo = -1)

int simpleMessageBox (QString text, int buttons = (int)MessageBoxButton::OK, QString icon = QString(), int defaultbutton = (MessageBoxButton)0, int cancelbutton = (MessageBoxButton)0)

~ActionExecutionUserFeedback ( )

class Choice
    #include <actionexecutionuserfeedback.h> A data structure for a choice (in an GridformEntry).

```

## Public Functions

```

Choice (QString label, QString text = QString())

```

## Public Members

```

QString label
QString text

```

```

class Choices
    #include <actionexecutionuserfeedback.h> A data structure for a list of Choice instances.

```

### Public Members

QList<*Choice*\*> **list**

int **selected** = -1

**class GridformEntries**

*#include <actionexecutionuserfeedback.h>* A data structure for a list of entries in a simple chooser grid form.

### Public Functions

**GridformEntries** ()

Is intended to be directly constructed from everywhere.

**~GridformEntries** ()

*GridformEntry* \***newEntry** (QString *label*)

### Public Members

QList<*GridformEntry*\*> **list**

**class GridformEntry**

*#include <actionexecutionuserfeedback.h>* A data structure for an entry in a simple chooser grid form.

### Public Functions

**GridformEntry** (QString *label*, *Choices* *choices*)

Constructed only indirectly.

**~GridformEntry** ()

int **addChoice** (QString *label*, QString *text* = QString())

### Public Members

QString **label**

*Choices* **choices**

template<class **T**>

**class ActionFactory** : public *sh::actions::AbstractActionFactory*

*#include <actionfactory.h>* A typical implementation for an action factory.

See base class for more.



## Public Functions

**ActionFactory** (std::shared\_ptr<sh::actions::ActionCategory> *category*,  
 QList<std::shared\_ptr<Predicate>> *predicates*)  
 std::shared\_ptr<sh::actions::AbstractActionItem> **construct** (QList<std::shared\_ptr<sh::filesystem::FilesystemNode>>  
*nodes*)  
 std::shared\_ptr<ActionInstantiation> **actionAvailable** (ActionInstantiation \**instantiation*)

### class ActionInstantiation

#include <actionfactory.h> Holds various data around action creation.

Action creation is a process, which begins with querying all actions available for certain nodes. The action factories evaluate availabilities and store some construction information here. Many parties are involved in working with it. The created action, is stored in this object as well.

## Public Functions

**ActionInstantiation** (QList<std::shared\_ptr<sh::filesystem::FilesystemNode>> *selectedNodes* = {}, std::shared\_ptr<sh::filesystem::FilesystemNode>  
*currentDirectory* = nullptr)

Constructor for a typical fresh instance, containing infos about selected items and some more.

**ActionInstantiation** (const ActionInstantiation &*s*) = default

Clones an existing instance.

**ActionInstantiation** (const ActionInstantiation &*s*, std::shared\_ptr<sh::actions::AbstractActionItem>  
*createdAction*)

Clones an existing instance and sets createdAction. This is a convenience constructor used for some special cases.

## Public Members

QList<std::shared\_ptr<sh::filesystem::FilesystemNode>> **selectedNodes**

The selected nodes (initially: what's selected in the active file list).

std::shared\_ptr<sh::filesystem::FilesystemNode> **currentDirectory**

The current directory (which is visible in the active file list).

bool **resolveLinks** = true

If links shall be resolved for evaluation and action creation.

bool **isLookupOnCurrentDirectoryLevel** = false

If this instantiation refers to a lookup for actions on 'current directory level' (instead of 'selection level').

AbstractActionFactory \***actionfactory** = 0

The action faction.

std::shared\_ptr<ActionCategory> **category** = 0

The action category.

std::shared\_ptr<AbstractActionItem> **createdAction** = 0

The created action.

QKeySequence **shortcut**

The requested keyboard shortcut for the action.

int **positionIndex** = 0  
The position information index.

**class ActionsManager** : public QObject, public *sh::base::Singleton*  
*#include <actionsmanager.h>* Factory for actions which are valid for a given list of FileSystemNode and tools for placing actions in some gui elements.

## Public Functions

void **getActions** (std::shared\_ptr<*ActionInstantiation*> *instantiation*,  
std::function<void> QList<std::shared\_ptr<*ActionInstantiation*>>,  
QStringList<std::shared\_ptr<*sh::actions::ActionInstantiation*>>  
> *callback*, std::function<void>std::shared\_ptr<*ActionInstantiation*>> *initializedcallback*  
= []) (std::shared\_ptr< *ActionInstantiation* >){} Asychonously gets actions for a list of  
*sh::filesystem::FileSystemNode*.

### Parameters

- *nodes*: The list of nodes.
- *callback*: This callback is called once the list is fetched.
- *initializedcallback*: This callback is called for each result action after initialization.

void **getDefaultAction** ( QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> *actionList*,  
std::function<void> std::shared\_ptr<*sh::actions::ActionActionItem*>  
> *callback* Determines the default action for a given list of actions.

### Parameters

- *actionList*: The list of all available actions.
- *callback*: This callback is called when the default action was found (with an already initialized action).

void **getActionForShortcut** ( QList<std::shared\_ptr<*filesystem::FileSystemNode*>> *se-*  
*lectedNodes*, std::shared\_ptr<*filesystem::FileSystemNode*>  
*folderNode*, int *key*, Qt::KeyboardModifiers *modifiers*,  
std::function<void> std::shared\_ptr<*sh::actions::AbstractActionItem*>  
> *callback* Asychonously gets the action for a shortcut.

### Parameters

- *selectedNodes*: The nodes which are currently selected in the active fileview.
- *folderNode*: The folder which is shown by the active fileview.
- *key*: The keypress character code.
- *modifiers*: The keypress modifiers.
- *callback*: This callback is called when an action was found (with an already initialized action).

void **registerActionFactory** (std::shared\_ptr<*sh::actions::AbstractActionFactory*> *action-*  
*Factory*)

Registers an action factory.

This makes an action implementation available in the toolbar and context menus.

QStringList<std::shared\_ptr<*sh::actions::ActionActionItem*>> **runningActions** ()  
Returns a list of all currently running actions.

void **doInitialize** ()  
Executes singleton initialization.

void **doShutdown** ()  
Executes singleton shutdown.

void **shutdown** ()  
 Shutdown down this singleton.

bool **isAlive** ()  
 Returns if this singleton is alive (true until its shutdown begins).

## Signals

void **runningActionsChanged** ()  
 Emitted when the list of running actions changed.

## Public Static Functions

QString **selectionLabel** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> selection)  
 Returns a short header label text for a node selection.

QString **directoryLabel** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> directory)  
 Returns a short header label text for a current directory.

void **sortAndSeparateActions** (QList<std::shared\_ptr<*ActionInstantiation*>> \*actions)  
 Sorts a lists of actionitems and adds separators between groups. Modifies the given list.

## Private Functions

**ActionsManager** ()

QList<std::shared\_ptr<*ActionInstantiation*>> **\_getActions\_raw** (std::shared\_ptr<*ActionInstantiation*> instantiation)

void **\_emit\_runningActionsChanged** ()

void **recursivelyInitializeAction** (std::shared\_ptr<*sh::actions::ActionInstantiation*> act-  
 sit, std::function<void> std::shared\_ptr<*sh::actions::ActionInstantiation*>  
 > callback, QList<std::shared\_ptr<*sh::actions::ActionInstantiation*>> sits = {}) Recursively initializes  
 a list of action items and calls a callback for each action item recursively afterwards.

**Parameters**

- **actions**: List of all action items to traverse.
- **callback**: This callback is called for each action item (even indirectly in a subtree hierarchy).

void **\_getDefaultAction\_helper** (QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>>  
 actionList, std::shared\_ptr<*GDAHstruct*> gdah)  
 Internal helper for the public *getDefaultAction()* method.

std::shared\_ptr<*sh::actions::ActionsManager::RunningActionsCounter*> **\_actionExecutionStarted** (std::shared\_ptr<*ac-*  
 tion

Helper for updating \_runningactions. .

### Private Members

```
QList<std::shared_ptr<sh::actions::AbstractActionFactory>> _actionfactories
QMutex _mutex_actionfactories
QList<std::shared_ptr<sh::actions::ActionActionItem>> _runningactions
QMutex _mutex_runningactions
```

### Friends

```
friend class ActionActionItem
struct GDAHstruct
    Internal data structure used for _getDefaultAction_helper();
```

### Public Functions

```
GDAHstruct () = default
GDAHstruct (const GDAHstruct &o) = default
```

### Public Members

```
sh::tools::AtomicCounter counter
std::function<void (std::shared_ptr<sh::actions::ActionActionItem>)> callback
int currentValue = 0
QList<std::shared_ptr<sh::actions::AbstractActionItem>> initialActionList
std::shared_ptr<sh::actions::AbstractActionItem> currentAction = 0
class RunningActionsCounter
    Helper for updating _runningactions. .
```

### Public Functions

```
RunningActionsCounter (ActionsManager *manager, std::shared_ptr<sh::actions::ActionActionItem>
                        action)
~RunningActionsCounter ()
```

### Private Members

```
ActionsManager *_manager
std::shared_ptr<sh::actions::ActionActionItem> _action
class ByRegExpPredicate : public sh::actions::Predicate
    #include <actionfactory.h> Shows actions only when the selection paths matches a regular expression.
```

## Public Functions

**ByRegExpPredicate** (QString *regexp*)

bool **evaluate** (*ActionInstantiation* \**instantiation*)

Evaluates if the predicate is solved.

If it returns `true`, the evaluation proceeds and eventually succeeds. If `false`, it stops at this place.

void **prepare** (*ActionInstantiation* \**instantiation*)

Prepares the evaluation.

## Private Members

QRegExp **filterRegExp**

**class DontResolveLinksPredicate** : public *sh::actions::Predicate*  
*#include <actionfactory.h>* Disables links resolving.

## Public Functions

**DontResolveLinksPredicate** ()

void **prepare** (*ActionInstantiation* \**instantiation*)

Prepares the evaluation.

bool **evaluate** (*ActionInstantiation* \**instantiation*)

Evaluates if the predicate is solved.

If it returns `true`, the evaluation proceeds and eventually succeeds. If `false`, it stops at this place.

**class HeaderActionItem** : public *sh::actions::AbstractActionItem*  
*#include <headeractionitem.h>* A header.

Use it as all the other *AbstractActionItem* classes. It will lead to a visual header in the parent menu.

## Public Functions

**HeaderActionItem** (QString *text*, QString *icon* = QString())

Is intended to be directly constructed from everywhere.

QString **text** ()

Returns the displayed text for this action.

QString **icon** ()

Returns the icon for this action.

bool **enabled** ()

Checks if this action is enabled.

bool **isChecked** ()

Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () const

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

```
void setText (QString text)  
    Sets the displayed text.  
  
void setIcon (QString icon)  
    Sets the icon.  
  
void setEnabled (bool enabled)  
    Sets if the item is enabled.  
  
void setChecked (bool checked)  
    Sets if the item is checked (has a cross in the ui).  
  
void setVisible (bool visible)  
    Sets the visibility of this item.  
  
bool visible ()  
    Checks the visibility of this item (non-recursively).  
  
std::weak_ptr<AbstractActionItem> parentAction ()  
    Returns the parent action, if it is added to a container.  
  
void __setParentAction (std::shared_ptr<AbstractActionItem> parent)  
    Sets the parent action. .  
  
void initializeAsync (std::function<void>  
    > oninitialized = 0) Asynchronously initializes the action.  
  
void initializeSync ()  
    Synchronously initializes the action.  
  
bool isInitialized ()  
    Checks if this action is initialized.  
  
bool isInitializing ()  
    Checks if this action is initializing.
```

## Signals

```
void changed ()  
    Emits when some data changed.
```

```
class HideOnCurrentDirectoryLevelPredicate : public sh::actions::Predicate  
    #include <actionfactory.h> Shows actions only when not searching for ‘current directory level’ actions.
```

## Public Functions

```
HideOnCurrentDirectoryLevelPredicate ()
```

```
bool evaluate (ActionInstantiation *instantiation)  
    Evaluates if the predicate is solved.
```

If it returns `true`, the evaluation proceeds and eventually succeeds. If `false`, it stops at this place.

```
void prepare (ActionInstantiation *instantiation)  
    Prepares the evaluation.
```

```
class HideOnSelectionLevelPredicate : public sh::actions::Predicate  
    #include <actionfactory.h> Shows actions only when not searching for ‘selection level’ actions.
```

## Public Functions

**HideOnSelectionLevelPredicate** ()

bool **evaluate** (*ActionInstantiation* \*instantiation)

Evaluates if the predicate is solved.

If it returns `true`, the evaluation proceeds and eventually succeeds. If `false`, it stops at this place.

void **prepare** (*ActionInstantiation* \*instantiation)

Prepares the evaluation.

**class KeyShortcutPredicate** : public *sh::actions::Predicate*  
*#include <actionfactory.h>* Sets a keyboard shortcut.

## Public Functions

**KeyShortcutPredicate** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectoryLevel*)

void **prepare** (*ActionInstantiation* \*instantiation)

Prepares the evaluation.

bool **evaluate** (*ActionInstantiation* \*instantiation)

Evaluates if the predicate is solved.

If it returns `true`, the evaluation proceeds and eventually succeeds. If `false`, it stops at this place.

## Private Members

QKeySequence **shortcut**

bool **triggersOnCurrentDirectoryLevel**

**class OnDirectoriesPredicate** : public *sh::actions::Predicate*  
*#include <actionfactory.h>* Shows actions only on directories.

## Public Functions

**OnDirectoriesPredicate** ()

bool **evaluate** (*ActionInstantiation* \*instantiation)

Evaluates if the predicate is solved.

If it returns `true`, the evaluation proceeds and eventually succeeds. If `false`, it stops at this place.

void **prepare** (*ActionInstantiation* \*instantiation)

Prepares the evaluation.

**class OnFilesPredicate** : public *sh::actions::Predicate*  
*#include <actionfactory.h>* Shows actions only on files.

### Public Functions

**OnFilesPredicate** ( )

bool **evaluate** (*ActionInstantiation* \*instantiation)

Evaluates if the predicate is solved.

If it returns `true`, the evaluation proceeds and eventually succeeds. If `false`, it stops at this place.

void **prepare** (*ActionInstantiation* \*instantiation)

Prepares the evaluation.

**class OnLinksPredicate** : public *sh::actions::Predicate*  
*#include <actionfactory.h>* Shows actions only on links.

### Public Functions

**OnLinksPredicate** ( )

bool **evaluate** (*ActionInstantiation* \*instantiation)

Evaluates if the predicate is solved.

If it returns `true`, the evaluation proceeds and eventually succeeds. If `false`, it stops at this place.

void **prepare** (*ActionInstantiation* \*instantiation)

Prepares the evaluation.

**class OnSingleEntrySelectionPredicate** : public *sh::actions::Predicate*  
*#include <actionfactory.h>* Shows actions only on single-entry selections.

### Public Functions

**OnSingleEntrySelectionPredicate** ( )

bool **evaluate** (*ActionInstantiation* \*instantiation)

Evaluates if the predicate is solved.

If it returns `true`, the evaluation proceeds and eventually succeeds. If `false`, it stops at this place.

void **prepare** (*ActionInstantiation* \*instantiation)

Prepares the evaluation.

**class PositionIndexPredicate** : public *sh::actions::Predicate*  
*#include <actionfactory.h>* Sets a positioning information index.

### Public Functions

**PositionIndexPredicate** (int *positionIndex*)

void **prepare** (*ActionInstantiation* \*instantiation)

Prepares the evaluation.

bool **evaluate** (*ActionInstantiation* \*instantiation)

Evaluates if the predicate is solved.

If it returns `true`, the evaluation proceeds and eventually succeeds. If `false`, it stops at this place.



## Private Members

int **positionIndex**

### class Predicate

*#include <actionfactory.h>* Controls when and how an action is created in the action factory.

Subclassed by *sh::actions::ByRegExpPredicate*, *sh::actions::DontResolveLinksPredicate*,  
*sh::actions::HideOnCurrentDirectoryLevelPredicate*, *sh::actions::HideOnSelectionLevelPredicate*,  
*sh::actions::KeyShortcutPredicate*, *sh::actions::OnDirectoriesPredicate*, *sh::actions::OnFilesPredicate*,  
*sh::actions::OnLinksPredicate*, *sh::actions::OnSingleEntrySelectionPredicate*,  
*sh::actions::PositionIndexPredicate*

## Public Functions

**Predicate()**

void **prepare** (*ActionInstantiation \*instantiation*)

Prepares the evaluation.

bool **evaluate** (*ActionInstantiation \*instantiation*)

Evaluates if the predicate is solved.

If it returns `true`, the evaluation proceeds and eventually succeeds. If `false`, it stops at this place.

**~Predicate()**

### class SeparatorActionItem: public *sh::actions::AbstractActionItem*

*#include <separatoractionitem.h>* A separator.

Use it as all the other *AbstractActionItem* classes. It will lead to a visual separator in the parent menu.

## Public Functions

**SeparatorActionItem()**

Is intended to be directly constructed from everywhere.

QString **text** ()

Returns the displayed text for this action.

QString **icon** ()

Returns the icon for this action.

bool **enabled** ()

Checks if this action is enabled.

bool **isChecked** ()

Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See *ActionDefaultPrecedenceValues* for reference values.

```
void setText (QString text)  
    Sets the displayed text.  
  
void setIcon (QString icon)  
    Sets the icon.  
  
void setEnabled (bool enabled)  
    Sets if the item is enabled.  
  
void setChecked (bool checked)  
    Sets if the item is checked (has a cross in the ui).  
  
void setVisible (bool visible)  
    Sets the visibility of this item.  
  
bool visible ()  
    Checks the visibility of this item (non-recursively).  
  
std::weak_ptr<AbstractActionItem> parentAction ()  
    Returns the parent action, if it is added to a container.  
  
void __setParentAction (std::shared_ptr<AbstractActionItem> parent)  
    Sets the parent action. .  
  
void initializeAsync (std::function<void>  
    > oninitialized = 0) Asynchronously initializes the action.  
  
void initializeSync ()  
    Synchronously initializes the action.  
  
bool isInitialized ()  
    Checks if this action is initialized.  
  
bool isInitializing ()  
    Checks if this action is initializing.
```

## Signals

```
void changed ()  
    Emits when some data changed.
```

```
class SubmenuItem: public sh::actions::AbstractActionItem  
    #include <submenuactionitem.h> Abstract base class for a submenu.
```

It can be made visible in menus or the toolbar or executed directly.

Subclasses fill the submenu with other actions in order to be useful.

Subclassed by *sh::actions::common::ActionAbstractTransferTo*, *sh::actions::common::ActionBookmarkFolder*,  
*sh::actions::common::ActionGroups*, *sh::actions::common::ActionHistoryNavigate*,  
*sh::actions::common::ActionOpenFileWith*, *sh::actions::common::ActionOpenTerminal*,  
*sh::actions::mainmenu::ActionApplyProfile*, *sh::actions::mainmenu::ActionMain*,  
*sh::actions::mainmenu::ActionNumberFileviews*, *sh::scripting::api::ApiSubmenuItem*

## Public Functions

**SubmenuItem** (QString *text*, const QList<std::shared\_ptr<sh::actions::AbstractActionItem>> *subitems* = {}, bool *enabled* = true, QString *icon* = QString(), int *defaultActionPrecedence* = 0, bool *checkable* = false, bool *isChecked* = false)

Is (for subclasses) intended to be directly constructed from everywhere or by registering a factory somewhere.

const QList<std::shared\_ptr<sh::actions::AbstractActionItem>> **subitems** ()

Returns the list of subitems from this submenu.

void **setSubitems** (const QList<std::shared\_ptr<sh::actions::AbstractActionItem>> *subitems*)

Sets the subitems.

QString **text** ()

Returns the displayed text for this action.

QString **icon** ()

Returns the icon for this action.

bool **enabled** ()

Checks if this action is enabled.

bool **isChecked** ()

Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () const

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<AbstractActionItem> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<AbstractActionItem> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>)

> *oninitialized* = 0 Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **subitemsChanged** ()  
Emits when the list of subitems changed.

void **changed** ()  
Emits when some data changed.

## Private Members

QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> **\_subitems**

### namespace common

Implementations of actions.

Subclasses of *sh::actions::AbstractActionItem* (and possibly some auxiliary stuff). Many action implementations used in Shallot are here. You can find other actions in the sibling namespaces and a few very special ones at different places.

**class ActionAbstractTransferTo : public *sh::actions::SubmenuActionItem***

*#include <actiontransferto.h>* Abstract action for transferring a selection to some other locations (e.g. to other file views, bookmarks).

Subclassed by *sh::actions::common::ActionTransferToCopy*, *sh::actions::common::ActionTransferToMove*

## Public Functions

**ActionAbstractTransferTo** (QString *text*, QString *icon*,  
QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>>  
*nodes*)

**const** QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> **subitems** ()  
Returns the list of subitems from this submenu.

void **setSubitems** (**const** QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>>  
*subitems*)  
Sets the subitems.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckedable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () const

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>

> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **subitemsChanged** ()

Emits when the list of subitems changed.

void **changed** ()

Emits when some data changed.

## Friends

**friend class** ActionTransferToHelper

**class** ActionAbstractTransferTree : public [sh::actions::ActionActionItem](#)

*#include <actionabstracttransfertree.h>* Abstract action for transferring (copying, moving, et al) a tree of items.

Subclassed by [sh::actions::common::ActionCopyTree](#)

## Public Functions

**ActionAbstractTransferTree** (QList<std::shared\_ptr<const [sh::filesystem::Eurl](#)>>  
sources, std::shared\_ptr<const [sh::filesystem::Eurl](#)>  
destination)

**~ActionAbstractTransferTree** ()

void **execute** ()  
Executes this action.

void **execute** ([sh::actions::ActionExecutionInfo](#) \*info)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence shortcut, bool triggersOnCurrentDirectory = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString text)  
Sets the displayed text.

void **setIcon** (QString icon)  
Sets the icon.

```

void setEnabled (bool enabled)
    Sets if the item is enabled.

void setChecked (bool checked)
    Sets if the item is checked (has a cross in the ui).

void setVisible (bool visible)
    Sets the visibility of this item.

bool visible ()
    Checks the visibility of this item (non-recursively).

std::weak_ptr<AbstractActionItem> parentAction ()
    Returns the parent action, if it is added to a container.

void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.

```

## Signals

```

void changed ()
    Emits when some data changed.

```

## Public Static Functions

```

void makereadableunits (double *value, QString *vunit)

```

```

class MyFilesystemOperationProgressMonitor : public sh::filesystem::FilesystemOperationProgress
    #include <actionabstracttransfertree.h>

```

## Public Functions

```

MyFilesystemOperationProgressMonitor (ActionExecutionInfo *info)

```

```

bool hasItemInfo ()
    Checks if this progress monitor provides information about how many items of a certain total
    number are transferred so far.

quint64 doneItems ()
    Checks how many items are transferred so far.

quint64 allItems ()
    Checks how many items are to be transferred in total (as predicted in the current moment).

bool hasBytesInfo ()
    Checks if this progress monitor provides information about how many byte of a certain total
    number are transferred so far.

```

quint64 **doneBytes** ()

Checks how many bytes are transferred so far.

quint64 **allBytes** ()

Checks how many bytes are to be transferred in total (as predicted in the current moment).

QString **getItemInfoFrom** ()

Returns the current source of transfer (as textual information).

QString **getItemInfoTo** ()

Returns the current destination of transfer (as textual information).

QString **estimation** ()

Returns the current performance and time estimation (as textual information).

**class ActionAddToBookmarks : public [sh::actions::ActionActionItem](#)**  
*#include <actionbookmarks.h>* Action for bookmarking a directory.

### Public Functions

**ActionAddToBookmarks** (QList<std::shared\_ptr<[sh::filesystem::FilesystemNode](#)>> *nodes*)

void **execute** ()

Executes this action.

void **execute** ([sh::actions::ActionExecutionInfo](#) \**info*)

Executes this action.

QKeySequence **shortcuthint** ()

Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()

Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)

Sets the keyboard shortcut for triggering this action.

QString **text** ()

Returns the displayed text for this action.

QString **icon** ()

Returns the icon for this action.

bool **enabled** ()

Checks if this action is enabled.

bool **isChecked** ()

Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.



```

void setIcon (QString icon)
    Sets the icon.

void setEnabled (bool enabled)
    Sets if the item is enabled.

void setChecked (bool checked)
    Sets if the item is checked (has a cross in the ui).

void setVisible (bool visible)
    Sets the visibility of this item.

bool visible ()
    Checks the visibility of this item (non-recursively).

std::weak_ptr<AbstractActionItem> parentAction ()
    Returns the parent action, if it is added to a container.

void __setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.

```

## Signals

```

void changed ()
    Emits when some data changed.

```

## Public Static Functions

```

void doInitialize ()

void doShutdown ()

```

```

class ActionBookmark : public sh::actions::ActionActionItem
    #include <actionbookmarks.h> Action for navigating to a certain bookmark.

```

## Public Functions

```

ActionBookmark (QString label, std::shared_ptr<const sh::filesystem::Eurl> eurl)

void execute ()
    Executes this action.

void execute (sh::actions::ActionExecutionInfo *info)
    Executes this action.

QKeySequence shortcuthint ()
    Returns the keyboard shortcut for triggering this action.

```

bool **shortcutHintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcutHint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

**class ActionBookmarkFolder**: public *sh::actions::SubmenuItem*  
*#include <actionbookmarks.h>* Submenu action of bookmarks.  
Subclassed by *sh::actions::common::ActionBookmarks*

## Public Functions

**ActionBookmarkFolder** (QString *label*)

void **addSubitem** (std::shared\_ptr<*sh::actions::AbstractActionItem*> *item*)

void **clearSubitems** ()

**const** QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> **subitems** ()  
Returns the list of subitems from this submenu.

void **setSubitems** (**const**           QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>>  
                                  *subitems*)  
Sets the subitems.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [\*ActionDefaultPrecedenceValues\*](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<*AbstractActionItem*> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<*AbstractActionItem*> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **subitemsChanged** ()  
Emits when the list of subitems changed.

void **changed** ()  
Emits when some data changed.

**class ActionBookmarks : public *sh::actions::common::ActionBookmarkFolder***  
*#include <actionbookmarks.h>* Bookmarking action (the main one you see in the toolbar)

## Public Functions

**ActionBookmarks** ()

void **initialize** ()  
Initialize the action. This should make the time-consuming parts, e.g. for determining a label or enabled state.

void **addSubitem** (std::shared\_ptr<*sh::actions::AbstractActionItem*> *item*)

void **clearSubitems** ()

**const** QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> **subitems** ()  
Returns the list of subitems from this submenu.

void **setSubitems** (**const**           QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>>  
                                  *subitems*)  
Sets the subitems.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
 Checks if this action is enabled.

bool **isChecked** ()  
 Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
 Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
 Returns the precedence for becoming the default action of a parent submenu.  
 This e.g. leads to a bold label.  
 The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
 Sets the displayed text.

void **setIcon** (QString *icon*)  
 Sets the icon.

void **setEnabled** (bool *enabled*)  
 Sets if the item is enabled.

void **setChecked** (bool *checked*)  
 Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
 Sets the visibility of this item.

bool **visible** ()  
 Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
 Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
 Sets the parent action. .

void **initializeAsync** (std::function<void>  
 > *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
 Synchronously initializes the action.

bool **isInitialized** ()  
 Checks if this action is initialized.

bool **isInitializing** ()  
 Checks if this action is initializing.

## Signals

void **subitemsChanged** ()  
Emits when the list of subitems changed.

void **changed** ()  
Emits when some data changed.

**class ActionClipboardCopy** : public *sh::actions::ActionActionItem*  
*#include <actionclipboard.h>* Action for copying to clipboard.

## Public Functions

**ActionClipboardCopy** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> *nodes*)

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcutHint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [\*ActionDefaultPrecedenceValues\*](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

```

void setChecked (bool checked)
    Sets if the item is checked (has a cross in the ui).

void setVisible (bool visible)
    Sets the visibility of this item.

bool visible ()
    Checks the visibility of this item (non-recursively).

std::weak_ptr<AbstractActionItem> parentAction ()
    Returns the parent action, if it is added to a container.

void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.

```

## Signals

```

void changed ()
    Emits when some data changed.

```

## Public Static Functions

```

void doInitialize ()

void doShutdown ()

```

```

class ActionClipboardCut : public sh::actions::ActionActionItem
    #include <actionclipboard.h> Action for cutting to clipboard.

```

## Public Functions

```

ActionClipboardCut (QList<std::shared_ptr<sh::filesystem::FilesystemNode>> nodes)

```

```

void execute ()
    Executes this action.

```

```

void execute (sh::actions::ActionExecutionInfo *info)
    Executes this action.

```

```

QKeySequence shortcuthint ()
    Returns the keyboard shortcut for triggering this action.

```

```

bool shortcuthintTriggersOnCurrentDirectory ()
    Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the
    entry selection).

```

void **setShortcut** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.



## Signals

void **changed** ()  
Emits when some data changed.

## Public Static Functions

void **doInitialize** ()  
void **doShutdown** ()

**class ActionClipboardPaste** : public *sh::actions::ActionActionItem*  
*#include <actionclipboard.h>* Action for pasting from clipboard.

## Public Functions

**ActionClipboardPaste** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> *nodes*)

bool **shortcutTriggersOnFolder** ()

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See *ActionDefaultPrecedenceValues* for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<AbstractActionItem> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_\_setParentAction** (std::shared\_ptr<AbstractActionItem> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
    > *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

## Public Static Functions

void **doInitialize** ()  
void **doShutdown** ()

**class ActionClipboardPasteAsLink : public sh::actions::ActionActionItem**  
    *#include <actionclipboard.h>* Action for pasting from clipboard as link.

## Public Functions

**ActionClipboardPasteAsLink** (QList<std::shared\_ptr<sh::filesystem::FilesystemNode>>  
    *nodes*)

void **execute** ()  
Executes this action.

void **execute** (sh::actions::ActionExecutionInfo \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcutHintTriggersOnCurrentDirectory** ()  
 Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcutHint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
 Sets the keyboard shortcut for triggering this action.

QString **text** ()  
 Returns the displayed text for this action.

QString **icon** ()  
 Returns the icon for this action.

bool **enabled** ()  
 Checks if this action is enabled.

bool **isChecked** ()  
 Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
 Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () const  
 Returns the precedence for becoming the default action of a parent submenu.  
 This e.g. leads to a bold label.  
 The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
 Sets the displayed text.

void **setIcon** (QString *icon*)  
 Sets the icon.

void **setEnabled** (bool *enabled*)  
 Sets if the item is enabled.

void **setChecked** (bool *checked*)  
 Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
 Sets the visibility of this item.

bool **visible** ()  
 Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
 Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
 Sets the parent action. .

void **initializeAsync** (std::function<void>  
 > *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
 Synchronously initializes the action.

bool **isInitialized** ()  
 Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

## Public Static Functions

void **doInitialize** ()

void **doShutdown** ()

**class ActionCopyTree** : public *sh::actions::common::ActionAbstractTransferTree*  
*#include <actioncopytree.h>* Action copying a filesystem tree to some other place.  
Subclassed by *sh::actions::common::ActionMoveTree*

## Public Functions

**ActionCopyTree** (QList<std::shared\_ptr<const *sh::filesystem::Eurl*>> *sources*,  
std::shared\_ptr<const *sh::filesystem::Eurl*> *destination*)

void **action** (*sh::actions::ActionExecutionInfo* \**info*)  
The action implementation, i.e. what the actions should actually do.

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () const

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>

> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **changed** ()

Emits when some data changed.

## Public Static Functions

void **makereadableunits** (double *\*value*, QString *\*vunit*)

**class ActionCreateFile** : public [sh::actions::ActionActionItem](#)

*#include <actioncreatefile.h>* Action for creating a new file.

## Public Functions

**ActionCreateFile** (QList<std::shared\_ptr<[sh::filesystem::FilesystemNode](#)>> *nodes*)

void **execute** ()  
Executes this action.

void **execute** ([sh::actions::ActionExecutionInfo](#) \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

```

void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.

```

## Signals

```

void changed ()
    Emits when some data changed.

```

## Public Static Functions

```

void doInitialize ()

void doShutdown ()

```

```

class ActionCreateFolder : public sh::actions::ActionActionItem
    #include <actioncreatefolder.h> Action for creating a new directory.

```

## Public Functions

```

ActionCreateFolder (QList<std::shared_ptr<sh::filesystem::FilesystemNode>> nodes)

bool shortcutTriggersOnFolder ()

void execute ()
    Executes this action.

void execute (sh::actions::ActionExecutionInfo *info)
    Executes this action.

QKeySequence shortcuthint ()
    Returns the keyboard shortcut for triggering this action.

bool shortcuthintTriggersOnCurrentDirectory ()
    Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the
    entry selection).

void setShortcuthint (QKeySequence shortcut, bool triggersOnCurrentDirectory = false)
    Sets the keyboard shortcut for triggering this action.

QString text ()
    Returns the displayed text for this action.

QString icon ()
    Returns the icon for this action.

bool enabled ()
    Checks if this action is enabled.

```

bool **isChecked** ()

Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>)

> *oninitialized* = 0 Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **changed** ()

Emits when some data changed.



## Public Static Functions

void **doInitialize** ()

void **doShutdown** ()

**class ActionDeleteItems : public *sh::actions::ActionActionItem***  
*#include <actiondeleteitems.h>* Action deleting a list of filesystem items.

## Public Functions

**ActionDeleteItems** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> *nodes*)

void **execute** ()

Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)

Executes this action.

QKeySequence **shortcuthint** ()

Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()

Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcut** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)

Sets the keyboard shortcut for triggering this action.

QString **text** ()

Returns the displayed text for this action.

QString **icon** ()

Returns the icon for this action.

bool **enabled** ()

Checks if this action is enabled.

bool **isChecked** ()

Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [\*ActionDefaultPrecedenceValues\*](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<*AbstractActionItem*> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<*AbstractActionItem*> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

## Public Static Functions

void **doInitialize** ()  
void **doShutdown** ()

**class ActionExecute** : public *sh::actions::ActionActionItem*  
*#include <actionexecute.h>* Action executing a file (if executable).

## Public Functions

**ActionExecute** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> *nodes*)

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

**QString text ()**  
Returns the displayed text for this action.

**QString icon ()**  
Returns the icon for this action.

**bool enabled ()**  
Checks if this action is enabled.

**bool isChecked ()**  
Checks if this action is checked (has a cross in the ui).

**bool isCheckable ()**  
Checks if this action is checkable (can have a cross in the ui).

**int defaultActionPrecedence () const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

**void setText (QString text)**  
Sets the displayed text.

**void setIcon (QString icon)**  
Sets the icon.

**void setEnabled (bool enabled)**  
Sets if the item is enabled.

**void setChecked (bool checked)**  
Sets if the item is checked (has a cross in the ui).

**void setVisible (bool visible)**  
Sets the visibility of this item.

**bool visible ()**  
Checks the visibility of this item (non-recursively).

**std::weak\_ptr<AbstractActionItem> parentAction ()**  
Returns the parent action, if it is added to a container.

**void \_setParentAction (std::shared\_ptr<AbstractActionItem> parent)**  
Sets the parent action. .

**void initializeAsync (std::function<void>  
> oninitialized = 0)**Asynchronously initializes the action.

**void initializeSync ()**  
Synchronously initializes the action.

**bool isInitialized ()**  
Checks if this action is initialized.

**bool isInitializing ()**  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

## Public Static Functions

void **doInitialize** ()

void **doShutdown** ()

**class ActionGroups** : public *sh::actions::SubmenuItem*  
*#include <actiongroups.h>* Action for showing registered actions grouped by an *sh::actions::ActionCategory*.

## Public Functions

**ActionGroups** (std::shared\_ptr<*sh::actions::ActionCategory*> category)

void **setGroupActions** (std::shared\_ptr<*sh::actions::ActionInstantiation*> rootsituation,  
QList<std::shared\_ptr<*sh::actions::ActionInstantiation*>> selacts,  
QList<std::shared\_ptr<*sh::actions::ActionInstantiation*>> directs)

const QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> **subitems** ()  
Returns the list of subitems from this submenu.

void **setSubitems** (const QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>>  
subitems)  
Sets the subitems.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () const  
Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString text)  
Sets the displayed text.

void **setIcon** (QString icon)  
Sets the icon.

void **setEnabled** (bool enabled)  
Sets if the item is enabled.

```

void setChecked (bool checked)
    Sets if the item is checked (has a cross in the ui).

void setVisible (bool visible)
    Sets the visibility of this item.

bool visible ()
    Checks the visibility of this item (non-recursively).

std::weak_ptr<AbstractActionItem> parentAction ()
    Returns the parent action, if it is added to a container.

void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.

```

## Signals

```

void subitemsChanged ()
    Emits when the list of subitems changed.

void changed ()
    Emits when some data changed.

```

```

class ActionHistoryNavigate : public sh::actions::SubmenuItem
    #include <actionnavigation.h> Abstract action for navigating in the directory history stack (stepwise
    or randomly).

    Subclassed          by          sh::actions::common::ActionHistoryNavigateBackward,
    sh::actions::common::ActionHistoryNavigateForward

```

## Public Functions

```

ActionHistoryNavigate (QString text, std::function<QList<sh::tools::HistoryTracker<std::shared_ptr<const
    sh::filesystem::Eurl>>::HistoryEntry>>
    > _getentriesfctQString icon

const QList<std::shared_ptr<sh::actions::AbstractActionItem>> subitems ()
    Returns the list of subitems from this submenu.

void setSubitems (const          QList<std::shared_ptr<sh::actions::AbstractActionItem>>
    subitems)
    Sets the subitems.

QString text ()
    Returns the displayed text for this action.

QString icon ()
    Returns the icon for this action.

```

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **subitemsChanged** ()  
Emits when the list of subitems changed.

void **changed** ()  
Emits when some data changed.

**class ActionHistoryNavigateBackward**: public [sh::actions::common::ActionHistoryNavigate](#)  
*#include <actionnavigation.h>* Action for navigating backward in the directory history stack.

## Public Functions

**ActionHistoryNavigateBackward** ()

const QList<std::shared\_ptr<[sh::actions::AbstractActionItem](#)>> **subitems** ()  
Returns the list of subitems from this submenu.

void **setSubitems** (const QList<std::shared\_ptr<[sh::actions::AbstractActionItem](#)>>  
subitems)  
Sets the subitems.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () const  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString text)  
Sets the displayed text.

void **setIcon** (QString icon)  
Sets the icon.

void **setEnabled** (bool enabled)  
Sets if the item is enabled.

void **setChecked** (bool checked)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool visible)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

`std::weak_ptr<AbstractActionItem> parentAction ()`  
Returns the parent action, if it is added to a container.

`void _setParentAction (std::shared_ptr<AbstractActionItem> parent)`  
Sets the parent action. .

`void initializeAsync (std::function<void>  
> oninitialized = 0)` Asynchronously initializes the action.

`void initializeSync ()`  
Synchronously initializes the action.

`bool isInitialized ()`  
Checks if this action is initialized.

`bool isInitializing ()`  
Checks if this action is initializing.

## Signals

`void subitemsChanged ()`  
Emits when the list of subitems changed.

`void changed ()`  
Emits when some data changed.

**class ActionHistoryNavigateForward**: public *sh::actions::common::ActionHistoryNavigate*  
*#include <actionnavigation.h>* Action for navigating forward in the directory history stack.

## Public Functions

**ActionHistoryNavigateForward ()**

**const** `QList<std::shared_ptr<sh::actions::AbstractActionItem>>` **subitems ()**  
Returns the list of subitems from this submenu.

`void setSubitems (const QList<std::shared_ptr<sh::actions::AbstractActionItem>>  
subitems)`  
Sets the subitems.

`QString text ()`  
Returns the displayed text for this action.

`QString icon ()`  
Returns the icon for this action.

`bool enabled ()`  
Checks if this action is enabled.

`bool isChecked ()`  
Checks if this action is checked (has a cross in the ui).

`bool isCheckable ()`  
Checks if this action is checkable (can have a cross in the ui).

`int defaultActionPrecedence () const`  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.



The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

```
void setText (QString text)
    Sets the displayed text.

void setIcon (QString icon)
    Sets the icon.

void setEnabled (bool enabled)
    Sets if the item is enabled.

void setChecked (bool checked)
    Sets if the item is checked (has a cross in the ui).

void setVisible (bool visible)
    Sets the visibility of this item.

bool visible ()
    Checks the visibility of this item (non-recursively).

std::weak_ptr<AbstractActionItem> parentAction ()
    Returns the parent action, if it is added to a container.

void __setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.
```

## Signals

```
void subitemsChanged ()
    Emits when the list of subitems changed.

void changed ()
    Emits when some data changed.

class ActionManageBookmarks : public sh::actions::ActionActionItem
    #include <actionbookmarks.h> Action for managing bookmarks.
```

## Public Functions

**ActionManageBookmarks ()**

void **execute ()**  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo \*info*)  
Executes this action.

QKeySequence **shortcuthint ()**  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory ()**  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text ()**  
Returns the displayed text for this action.

QString **icon ()**  
Returns the icon for this action.

bool **enabled ()**  
Checks if this action is enabled.

bool **isChecked ()**  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable ()**  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence () const**  
Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible ()**  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction ()**  
Returns the parent action, if it is added to a container.

```

void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action.

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.

```

## Signals

```

void changed ()
    Emits when some data changed.

```

```

class ActionMoveTree: public sh::actions::common::ActionCopyTree
    #include <actionmovetree.h> Action moving a filesystem tree to some other place.

```

## Public Functions

```

ActionMoveTree (QList<std::shared_ptr<const sh::filesystem::Eurl>> sources,
    std::shared_ptr<const sh::filesystem::Eurl> destination)

void action (sh::actions::ActionExecutionInfo *info)
    The action implementation, i.e. what the actions should actually do.

void execute ()
    Executes this action.

void execute (sh::actions::ActionExecutionInfo *info)
    Executes this action.

QKeySequence shortcuthint ()
    Returns the keyboard shortcut for triggering this action.

bool shortcuthintTriggersOnCurrentDirectory ()
    Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the
    entry selection).

void setShortcuthint (QKeySequence shortcut, bool triggersOnCurrentDirectory = false)
    Sets the keyboard shortcut for triggering this action.

QString text ()
    Returns the displayed text for this action.

QString icon ()
    Returns the icon for this action.

bool enabled ()
    Checks if this action is enabled.

bool isChecked ()
    Checks if this action is checked (has a cross in the ui).

bool isCheckable ()
    Checks if this action is checkable (can have a cross in the ui).

```

int **defaultActionPrecedence** () const

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>

> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **changed** ()

Emits when some data changed.

## Public Static Functions

void **makeReadableUnits** (double *\*value*, QString *\*vunit*)

**class ActionNavigateInHistory** : public [sh::actions::ActionActionItem](#)

*#include <actionnavigation.h>* Action for navigating to one particular place in the directory history stack.

## Public Functions

**ActionNavigateInHistory** (*sh::ui::FileView* \*filelist, int idx, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, bool isdefault)

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \*info)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence shortcut, bool triggersOnCurrentDirectory = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () const  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString text)  
Sets the displayed text.

void **setIcon** (QString icon)  
Sets the icon.

void **setEnabled** (bool enabled)  
Sets if the item is enabled.

void **setChecked** (bool checked)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool visible)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<*AbstractActionItem*> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>

> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **changed** ()

Emits when some data changed.

**class ActionOpenArchive : public *sh::actions::ActionActionItem***

*#include <actionopenarchive.h>* Action opening a share archive (by creating a root node for it).

Subclassed by *sh::filesystemhandlers::ArchiveFilesystemHandler::ActionOpenTarArchive*,  
*sh::filesystemhandlers::ArchiveFilesystemHandler::ActionOpenZipArchive*

## Public Functions

**ActionOpenArchive** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> *nodes*)

void **execute** ()

Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)

Executes this action.

QKeySequence **shortcuthint** ()

Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()

Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)

Sets the keyboard shortcut for triggering this action.

QString **text** ()

Returns the displayed text for this action.

QString **icon** ()

Returns the icon for this action.

bool **enabled** ()

Checks if this action is enabled.

bool **isChecked** ()

Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () const

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>

> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **changed** ()

Emits when some data changed.

**class ActionOpenDirectoryInNewWindow** : public [sh::actions::ActionActionItem](#)

*#include <actionopendirectoryinnewwindow.h>* Action for opening a file with a automatically chosen program.

## Public Functions

**ActionOpenDirectoryInNewWindow** (QList<std::shared\_ptr<[sh::filesystem::FilesystemNode](#)>>  
nodes)

void **execute** ()  
Executes this action.

void **execute** ([sh::actions::ActionExecutionInfo](#) \*info)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence shortcut, bool triggersOnCurrentDirectory = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () const  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString text)  
Sets the displayed text.

void **setIcon** (QString icon)  
Sets the icon.

void **setEnabled** (bool enabled)  
Sets if the item is enabled.

void **setChecked** (bool checked)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool visible)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.



```

void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.

```

## Signals

```

void changed ()
    Emits when some data changed.

```

## Public Static Functions

```

void doInitialize ()

void doShutdown ()

```

```

class ActionOpenFile : public sh::actions::ActionActionItem
    #include <actionopenfile.h> Action for opening a file with a automatically chosen program.

```

## Public Functions

```

ActionOpenFile (QList<std::shared_ptr<sh::filesystem::FilesystemNode>> nodes)

void execute ()
    Executes this action.

void execute (sh::actions::ActionExecutionInfo *info)
    Executes this action.

QKeySequence shortcuthint ()
    Returns the keyboard shortcut for triggering this action.

bool shortcuthintTriggersOnCurrentDirectory ()
    Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the
    entry selection).

void setShortcuthint (QKeySequence shortcut, bool triggersOnCurrentDirectory = false)
    Sets the keyboard shortcut for triggering this action.

QString text ()
    Returns the displayed text for this action.

QString icon ()
    Returns the icon for this action.

bool enabled ()
    Checks if this action is enabled.

```

bool **isChecked** ()

Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>)

> *oninitialized* = 0 Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **changed** ()

Emits when some data changed.

## Public Static Functions

void **doInitialize** ()

void **doShutdown** ()

**class ActionOpenFileWith**: public *sh::actions::SubmenuItem*  
*#include <actionopenfilewith.h>* Submenu action for opening a file with a manually chosen program.

## Public Functions

**ActionOpenFileWith** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> *nodes*)

**const** QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> **subitems** ()  
 Returns the list of subitems from this submenu.

void **setSubitems** (**const** QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>>  
*subitems*)  
 Sets the subitems.

QString **text** ()  
 Returns the displayed text for this action.

QString **icon** ()  
 Returns the icon for this action.

bool **enabled** ()  
 Checks if this action is enabled.

bool **isChecked** ()  
 Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
 Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
 Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
 Sets the displayed text.

void **setIcon** (QString *icon*)  
 Sets the icon.

void **setEnabled** (bool *enabled*)  
 Sets if the item is enabled.

void **setChecked** (bool *checked*)  
 Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
 Sets the visibility of this item.

bool **visible** ()  
 Checks the visibility of this item (non-recursively).

std::weak\_ptr<*AbstractActionItem*> **parentAction** ()  
 Returns the parent action, if it is added to a container.

```
void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.
```

## Signals

```
void subitemsChanged ()
    Emits when the list of subitems changed.

void changed ()
    Emits when some data changed.
```

## Public Static Functions

```
void waitProgramClosedIfNeeded (sh::actions::ActionExecutionInfo *info, QStringList
    filelist)

void doInitialize ()

void doShutdown ()
```

## Friends

```
friend class ActionOpenFile

class _ActionOpenFileWithEntry : public sh::actions::ActionActionItem
```

## Public Functions

```
_ActionOpenFileWithEntry (QList<std::shared_ptr<sh::filesystem::FilesystemNode>>
    nodes, QString name, QString cmd, QStringList argu-
    ments, QStringList rememberformimetype)

void action (sh::actions::ActionExecutionInfo *info)
    The action implementation, i.e. what the actions should actually do.

void execute ()
    Executes this action.

void execute (sh::actions::ActionExecutionInfo *info)
    Executes this action.

QKeySequence shortcuthint ()
    Returns the keyboard shortcut for triggering this action.
```

bool **shortcutHintTriggersOnCurrentDirectory** ()  
 Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcutHint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
 Sets the keyboard shortcut for triggering this action.

QString **text** ()  
 Returns the displayed text for this action.

QString **icon** ()  
 Returns the icon for this action.

bool **enabled** ()  
 Checks if this action is enabled.

bool **isChecked** ()  
 Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
 Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
 Returns the precedence for becoming the default action of a parent submenu.  
 This e.g. leads to a bold label.  
 The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
 Sets the displayed text.

void **setIcon** (QString *icon*)  
 Sets the icon.

void **setEnabled** (bool *enabled*)  
 Sets if the item is enabled.

void **setChecked** (bool *checked*)  
 Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
 Sets the visibility of this item.

bool **visible** ()  
 Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
 Returns the parent action, if it is added to a container.

void **\_\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
 Sets the parent action. .

void **initializeAsync** (std::function<void> > *oninitialized* = 0)  
 Asynchronously initializes the action.

void **initializeSync** ()  
 Synchronously initializes the action.

bool **isInitialized** ()  
 Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Public Members

QString **\_command**  
QStringList **\_commandArguments**  
QList<std::shared\_ptr<sh::filesystem::FileSystemNode>> **\_nodes**

## Signals

void **changed** ()  
Emits when some data changed.

class **\_ActionOpenFileWithUI** : public *sh::actions::ActionActionItem*

## Public Functions

**\_ActionOpenFileWithUI** (QList<std::shared\_ptr<sh::filesystem::FileSystemNode>>  
nodes, QString mimetype)

void **action** (*sh::actions::ActionExecutionInfo* \*info)  
The action implementation, i.e. what the actions should actually do.

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \*info)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence shortcut, bool triggersOnCurrentDirectory =  
false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () const

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>)

> *oninitialized* = 0 Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Public Members

QList<std::shared\_ptr<[sh::filesystem::FileSystemNode](#)>> **\_nodes**

QString **\_mimetype**

## Signals

void **changed** ()

Emits when some data changed.

**class ActionOpenSharcArchive : public [sh::actions::ActionActionItem](#)**

*#include <actionopensharcarchive.h>* Action opening a sharc archive (by creating a root node for it).

## Public Functions

**ActionOpenSharcArchive** (QList<std::shared\_ptr<[sh::filesystem::FilesystemNode](#)>>  
nodes)

void **execute** ()  
Executes this action.

void **execute** ([sh::actions::ActionExecutionInfo](#) \*info)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence shortcut, bool triggersOnCurrentDirectory = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () const  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString text)  
Sets the displayed text.

void **setIcon** (QString icon)  
Sets the icon.

void **setEnabled** (bool enabled)  
Sets if the item is enabled.

void **setChecked** (bool checked)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool visible)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.



```

void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.

```

## Signals

```

void changed ()
    Emits when some data changed.

```

## Public Static Functions

```

void doInitialize ()

void doShutdown ()

```

```

class ActionOpenTerminal : public sh::actions::SubmenuActionItem
    #include <actionopenterminal.h> Action submenu with actions for opening commandline terminals.

```

## Public Functions

```

ActionOpenTerminal (QString dir)

void initialize ()
    Initialize the action. This should make the time-consuming parts, e.g. for determining a label or
    enabled state.

const QList<std::shared_ptr<sh::actions::AbstractActionItem>> subitems ()
    Returns the list of subitems from this submenu.

void setSubitems (const      QList<std::shared_ptr<sh::actions::AbstractActionItem>>
                  subitems)
    Sets the subitems.

QString text ()
    Returns the displayed text for this action.

QString icon ()
    Returns the icon for this action.

bool enabled ()
    Checks if this action is enabled.

bool isChecked ()
    Checks if this action is checked (has a cross in the ui).

bool isCheckable ()
    Checks if this action is checkable (can have a cross in the ui).

```

int **defaultActionPrecedence** () const

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>

> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **subitemsChanged** ()

Emits when the list of subitems changed.

void **changed** ()

Emits when some data changed.

**class ActionOpenTerminalAsRoot : public [sh::actions::common::ActionOpenTerminalAsUser](#)**  
*#include <actionopenterminal.h>* Action for opening a commandline terminal as root.

## Public Functions

**ActionOpenTerminalAsRoot** (QString *dir*)

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<*AbstractActionItem*> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

**class ActionOpenTerminalAsUser** : public *sh::actions::ActionActionItem*  
*#include <actionopenterminal.h>* Action for opening a commandline terminal (with current user).  
Subclassed by *sh::actions::common::ActionOpenTerminalAsRoot*

## Public Functions

**ActionOpenTerminalAsUser** (QString *dir*)

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () const

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>

> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **changed** ()

Emits when some data changed.

**class ActionRenameItem**: public [sh::actions::ActionActionItem](#)

*#include <actionrenameitem.h>* Action for renaming items.

## Public Functions

**ActionRenameItem** (QList<std::shared\_ptr<[sh::filesystem::FilesystemNode](#)>> *nodes*)

void **execute** ()  
Executes this action.

void **execute** ([sh::actions::ActionExecutionInfo](#) \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

```

void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.

```

## Signals

```

void changed ()
    Emits when some data changed.

```

## Public Static Functions

```

void doInitialize ()

void doShutdown ()

```

```

class ActionShowProperties : public sh::actions::ActionActionItem
    #include <actionshowproperties.h> Action for showing the properties dialog for filesystem nodes.

```

## Public Functions

```

ActionShowProperties (QList<std::shared_ptr<sh::filesystem::FilesystemNode>> nodes)

void execute ()
    Executes this action.

void execute (sh::actions::ActionExecutionInfo *info)
    Executes this action.

QKeySequence shortcuthint ()
    Returns the keyboard shortcut for triggering this action.

bool shortcuthintTriggersOnCurrentDirectory ()
    Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the
    entry selection).

void setShortcuthint (QKeySequence shortcut, bool triggersOnCurrentDirectory = false)
    Sets the keyboard shortcut for triggering this action.

QString text ()
    Returns the displayed text for this action.

QString icon ()
    Returns the icon for this action.

bool enabled ()
    Checks if this action is enabled.

```

bool **isChecked** ()

Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>)

> *oninitialized* = 0 Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **changed** ()

Emits when some data changed.



## Public Static Functions

void **doInitialize** ()

void **doShutdown** ()

**class ActionTransferToCopy** : public *sh::actions::common::ActionAbstractTransferTo*  
*#include <actiontransferto.h>* Action for copying a selection to some other locations.

## Public Functions

**ActionTransferToCopy** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> *nodes*)

**const** QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> **subitems** ()  
 Returns the list of subitems from this submenu.

void **setSubitems** (**const** QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>>  
*subitems*)  
 Sets the subitems.

QString **text** ()  
 Returns the displayed text for this action.

QString **icon** ()  
 Returns the icon for this action.

bool **enabled** ()  
 Checks if this action is enabled.

bool **isChecked** ()  
 Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
 Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
 Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
 Sets the displayed text.

void **setIcon** (QString *icon*)  
 Sets the icon.

void **setEnabled** (bool *enabled*)  
 Sets if the item is enabled.

void **setChecked** (bool *checked*)  
 Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
 Sets the visibility of this item.

bool **visible** ()  
 Checks the visibility of this item (non-recursively).

std::weak\_ptr<*AbstractActionItem*> **parentAction** ()  
 Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<*AbstractActionItem*> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **subitemsChanged** ()  
Emits when the list of subitems changed.

void **changed** ()  
Emits when some data changed.

## Public Static Functions

void **doInitialize** ()  
Only called by the Shallot infrastructure for initialization.

void **doShutdown** ()  
Only called by the Shallot infrastructure for initialization.

**class ActionTransferToHelper** : public *sh::actions::ActionActionItem*  
*#include <actiontransferto.h>* This action is used as subitems in *ActionAbstractTransferTo*.

## Public Functions

**ActionTransferToHelper** (QString *text*, QString *icon*, std::shared\_ptr<const  
*sh::filesystem::Eurl*> *transferto*,  
std::shared\_ptr<*ActionAbstractTransferTo*> *root*)

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

**class ActionTransferToMove** : public *sh::actions::common::ActionAbstractTransferTo*  
*#include <actiontransferto.h>* Action for moving a selection to some other locations.

## Public Functions

**ActionTransferToMove** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> *nodes*)

**const** QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> **subitems** ()  
Returns the list of subitems from this submenu.

void **setSubitems** (**const** QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>>  
*subitems*)  
Sets the subitems.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.  
The action with the highest value becomes the default. Only values >0 will be considered. See *ActionDefaultPrecedenceValues* for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<*AbstractActionItem*> **parentAction** ()  
Returns the parent action, if it is added to a container.

```
void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.
```

## Signals

```
void subitemsChanged ()
    Emits when the list of subitems changed.

void changed ()
    Emits when some data changed.
```

## Public Static Functions

```
void doInitialize ()

void doShutdown ()
```

### namespace mainmenu

Implementation of the main menu.

Subclasses of *sh::actions::AbstractActionItem* (and possibly some auxiliary stuff). You can find other actions in the sibling namespaces and a few very special ones at different places.

```
class ActionAbout : public sh::actions::ActionActionItem
    #include <actionabout.h> Action showing the Shallot about box.
```

## Public Functions

```
ActionAbout ()

void execute ()
    Executes this action.

void execute (sh::actions::ActionExecutionInfo *info)
    Executes this action.

QKeySequence shortcuthint ()
    Returns the keyboard shortcut for triggering this action.

bool shortcuthintTriggersOnCurrentDirectory ()
    Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the
    entry selection).

void setShortcuthint (QKeySequence shortcut, bool triggersOnCurrentDirectory = false)
    Sets the keyboard shortcut for triggering this action.
```

**QString text ()**  
Returns the displayed text for this action.

**QString icon ()**  
Returns the icon for this action.

**bool enabled ()**  
Checks if this action is enabled.

**bool isChecked ()**  
Checks if this action is checked (has a cross in the ui).

**bool isCheckedable ()**  
Checks if this action is checkable (can have a cross in the ui).

**int defaultActionPrecedence () const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

**void setText (QString text)**  
Sets the displayed text.

**void setIcon (QString icon)**  
Sets the icon.

**void setEnabled (bool enabled)**  
Sets if the item is enabled.

**void setChecked (bool checked)**  
Sets if the item is checked (has a cross in the ui).

**void setVisible (bool visible)**  
Sets the visibility of this item.

**bool visible ()**  
Checks the visibility of this item (non-recursively).

**std::weak\_ptr<AbstractActionItem> parentAction ()**  
Returns the parent action, if it is added to a container.

**void \_setParentAction (std::shared\_ptr<AbstractActionItem> parent)**  
Sets the parent action. .

**void initializeAsync (std::function<void>  
> oninitialized = 0)**Asynchronously initializes the action.

**void initializeSync ()**  
Synchronously initializes the action.

**bool isInitialized ()**  
Checks if this action is initialized.

**bool isInitializing ()**  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

**class ActionAbstractSelectNodes : public** [\*sh::actions::ActionActionItem\*](#)  
[\*#include <actionselect.h>\*](#) Abstract subclass for actions which select some nodes in the current file view.  
 Subclassed by [\*sh::actions::mainmenu::ActionInvertNodeSelection\*](#),  
[\*sh::actions::mainmenu::ActionSelectAllNodes\*](#)

## Public Functions

**ActionAbstractSelectNodes** (QString *text*)

void **execute** ()  
Executes this action.

void **execute** ([\*sh::actions::ActionExecutionInfo\*](#) \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [\*ActionDefaultPrecedenceValues\*](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

**class ActionApplyProfile** : public [sh::actions::SubmenuActionItem](#)  
*#include <actionapplyprofile.h>* Submenu action for switching the current profile.

## Public Functions

**ActionApplyProfile** ()

**const** QList<std::shared\_ptr<[sh::actions::AbstractActionItem](#)>> **subitems** ()  
Returns the list of subitems from this submenu.

void **setSubitems** (**const**           QList<std::shared\_ptr<[sh::actions::AbstractActionItem](#)>>  
                                  *subitems*)  
Sets the subitems.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).



bool **isCheckedable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () const

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>

> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **subitemsChanged** ()

Emits when the list of subitems changed.

void **changed** ()

Emits when some data changed.

class **ActionApplyProfileX**: public [sh::actions::ActionActionItem](#)

Action for switching the current profile to a certain one.

## Public Functions

**ActionApplyProfileX** (QString *profilename*, bool *checked*)

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

```

void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action.

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.

```

## Signals

```

void changed ()
    Emits when some data changed.

```

```

class ActionFileViewChangeIconDimension : public sh::actions::ActionActionItem
    #include <actionfileviewchangeicondimension.h> Action for changing the icon size.

Subclassed      by      sh::actions::mainmenu::ActionFileViewDecreaseIconDimension,
sh::actions::mainmenu::ActionFileViewIncreaseIconDimension

```

## Public Functions

```

ActionFileViewChangeIconDimension (QString name, int direction, QString icon,
    QKeySequence shortcut)

void execute ()
    Executes this action.

void execute (sh::actions::ActionExecutionInfo *info)
    Executes this action.

QKeySequence shortcuthint ()
    Returns the keyboard shortcut for triggering this action.

bool shortcuthintTriggersOnCurrentDirectory ()
    Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the
    entry selection).

void setShortcuthint (QKeySequence shortcut, bool triggersOnCurrentDirectory = false)
    Sets the keyboard shortcut for triggering this action.

QString text ()
    Returns the displayed text for this action.

QString icon ()
    Returns the icon for this action.

bool enabled ()
    Checks if this action is enabled.

bool isChecked ()
    Checks if this action is checked (has a cross in the ui).

bool isCheckable ()
    Checks if this action is checkable (can have a cross in the ui).

```

int **defaultActionPrecedence** () const

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>)

> *oninitialized* = 0 Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **changed** ()

Emits when some data changed.

**class ActionFileViewDecreaseIconDimension : public [sh::actions::mainmenu::ActionFileViewChangeIcon](#)**

*#include <actionfileviewchangeicondimension.h>* Action for decreasing the icon size.

## Public Functions

**ActionFileViewDecreaseIconDimension** ()

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo \*info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<*AbstractActionItem*> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>

> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **changed** ()

Emits when some data changed.

**class ActionFileviewFilesizeFormatting** : public *sh::actions::ActionActionItem*  
*#include <actionfileviewfilesizeformatting.h>* Action for toggling the filesize formatting mode for the current fileview.

## Public Functions

**ActionFileviewFilesizeFormatting** ()

void **execute** ()

Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)

Executes this action.

QKeySequence **shortcuthint** ()

Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()

Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)

Sets the keyboard shortcut for triggering this action.

QString **text** ()

Returns the displayed text for this action.

QString **icon** ()

Returns the icon for this action.

bool **enabled** ()

Checks if this action is enabled.

bool **isChecked** ()

Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () const

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>

> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **changed** ()

Emits when some data changed.

**class ActionFileViewIncreaseIconDimension : public [sh::actions::mainmenu::ActionFileViewChangeIcon](#)**

*#include <actionfileviewchangeicondimension.h>* Action for increasing the icon size.

## Public Functions

**ActionFileViewIncreaseIconDimension** ()

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo \*info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.



```
void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.
```

## Signals

```
void changed ()
    Emits when some data changed.
```

```
class ActionFileviewShowHiddenFiles : public sh::actions::ActionActionItem
    #include <actionfileviewshowhiddenfiles.h> Action which toggles the visibility of hidden files in the
    current fileview.
```

## Public Functions

```
ActionFileviewShowHiddenFiles ()

void execute ()
    Executes this action.

void execute (sh::actions::ActionExecutionInfo *info)
    Executes this action.

QKeySequence shortcuthint ()
    Returns the keyboard shortcut for triggering this action.

bool shortcuthintTriggersOnCurrentDirectory ()
    Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the
    entry selection).

void setShortcuthint (QKeySequence shortcut, bool triggersOnCurrentDirectory = false)
    Sets the keyboard shortcut for triggering this action.

QString text ()
    Returns the displayed text for this action.

QString icon ()
    Returns the icon for this action.

bool enabled ()
    Checks if this action is enabled.

bool isChecked ()
    Checks if this action is checked (has a cross in the ui).

bool isCheckable ()
    Checks if this action is checkable (can have a cross in the ui).
```

int **defaultActionPrecedence** () const

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>

> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **changed** ()

Emits when some data changed.

**class ActionFileviewView: public [sh::actions::ActionActionItem](#)**

*#include <actionfileviewview.h>* Action which toggles between icon- and listmode for the current fileview.

## Public Functions

**ActionFileviewView** ()

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo \*info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<*AbstractActionItem*> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>

> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **changed** ()

Emits when some data changed.

**class ActionGotoDirectory**: public *sh::actions::ActionActionItem*

*#include <actiongotodirectory.h>* Action which toggles the visibility of hidden files in the current fileview.

## Public Functions

**ActionGotoDirectory** ()

void **execute** ()

Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)

Executes this action.

QKeySequence **shortcuthint** ()

Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()

Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)

Sets the keyboard shortcut for triggering this action.

QString **text** ()

Returns the displayed text for this action.

QString **icon** ()

Returns the icon for this action.

bool **enabled** ()

Checks if this action is enabled.

bool **isChecked** ()

Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () const

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>

> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **changed** ()

Emits when some data changed.

**class ActionHelp** : public [sh::actions::ActionActionItem](#)

*#include <actionhelp.h>* Action for opening the Shallot help system.

## Public Functions

**ActionHelp** ()

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo \*info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

```

void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.

```

## Signals

```

void changed ()
    Emits when some data changed.

```

```

class ActionInvertNodeSelection : public sh::actions::mainmenu::ActionAbstractSelectNodes
    #include <actionselect.h> Action for inverting the node selection in the current file view.

```

## Public Functions

```

ActionInvertNodeSelection ()

void execute ()
    Executes this action.

void execute (sh::actions::ActionExecutionInfo *info)
    Executes this action.

QKeySequence shortcuthint ()
    Returns the keyboard shortcut for triggering this action.

bool shortcuthintTriggersOnCurrentDirectory ()
    Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the
    entry selection).

void setShortcuthint (QKeySequence shortcut, bool triggersOnCurrentDirectory = false)
    Sets the keyboard shortcut for triggering this action.

QString text ()
    Returns the displayed text for this action.

QString icon ()
    Returns the icon for this action.

bool enabled ()
    Checks if this action is enabled.

bool isChecked ()
    Checks if this action is checked (has a cross in the ui).

bool isCheckable ()
    Checks if this action is checkable (can have a cross in the ui).

int defaultActionPrecedence () const
    Returns the precedence for becoming the default action of a parent submenu.

```

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [\*ActionDefaultPrecedenceValues\*](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[\*AbstractActionItem\*](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_\_setParentAction** (std::shared\_ptr<[\*AbstractActionItem\*](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>)

> *oninitialized* = 0 Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **changed** ()

Emits when some data changed.

## Private Members

QList<std::shared\_ptr<[\*sh::filesystem::FilesystemNode\*](#)>> **wasselected**

**class ActionMain : public [\*sh::actions::SubmenuActionItem\*](#), public [\*sh::base::Singleton\*](#)**  
**#include <actionmain.h>** Submenu action providing the Shallot main menu.



## Public Functions

**const** QList<std::shared\_ptr<[sh::actions::AbstractActionItem](#)>> **subitems** ()  
Returns the list of subitems from this submenu.

void **setSubitems** (**const**           QList<std::shared\_ptr<[sh::actions::AbstractActionItem](#)>>  
                                  *subitems*)  
Sets the subitems.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
                      > *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

**bool isInitializing ()**  
Checks if this action is initializing.

**void doInitialize ()**  
Executes singleton initialization.

**void doShutdown ()**  
Executes singleton shutdown.

**void shutdown ()**  
Shutdown down this singleton.

**bool isAlive ()**  
Returns if this singleton is alive (true until its shutdown begins).

## Signals

**void subitemsChanged ()**  
Emits when the list of subitems changed.

**void changed ()**  
Emits when some data changed.

## Private Functions

**ActionMain ()**

**class ActionManageSavedSettings : public *sh::actions::ActionActionItem***  
*#include <actionmanagesavedsettings.h>* Action for opening the 'Manage saved settings' dialog.

## Public Functions

**ActionManageSavedSettings ()**

**void execute ()**  
Executes this action.

**void execute (*sh::actions::ActionExecutionInfo* \*info)**  
Executes this action.

**QKeySequence shortcuthint ()**  
Returns the keyboard shortcut for triggering this action.

**bool shortcuthintTriggersOnCurrentDirectory ()**  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

**void setShortcuthint (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)**  
Sets the keyboard shortcut for triggering this action.

**QString text ()**  
Returns the displayed text for this action.

**QString icon ()**  
Returns the icon for this action.

**bool enabled ()**  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
    > *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

**class ActionNumberFileviews : public [sh::actions::SubmenuActionItem](#)**  
    #include <actionnumberfileviews.h> Submenu action for changing the number of fileviews.

## Public Functions

## ActionNumberFileviews ()

**const** QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> **subitems** ()  
Returns the list of subitems from this submenu.

void **setSubitems** (const *subitems*)      QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>>  
 Sets the subitems.

QString **text** ()

Returns the displayed text for this action.

QString **icon** ()

Returns the icon for this action.

```
bool enabled()
```

Checks if this action is enabled.

bool **isChecked**()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

**int defaultActionPrecedence () const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

**void setIcon** (QString *icon*)  
Sets the icon.

**void setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

**void setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible**()  
Checks the visibility of this item (non-recursively).

`std::weak_ptr<AbstractActionItem> parentAction ()`  
Returns the parent action, if it is added to a container.

```
void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
```

Sets the parent action. .

```
void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.
```

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized**()  
Checks if this action is initialized.

bool **isInitializing**()  
Checks if this action is initializing.

## Signals

void **subitemsChanged**()  
Emits when the list of subitems changed.

void **changed**()  
Emits when some data changed.

## Friends

**friend class** ActionNumberFileviews\_Add

**friend class** ActionNumberFileviews\_Remove

**friend class** ActionNumberFileviews\_Remove\_Current

**class** ActionNumberFileviews\_Add: public *sh::actions::ActionActionItem*  
*#include <actionnumberfileviews.h>* Action for adding a new fileview.

## Public Functions

**ActionNumberFileviews\_Add**(QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>>  
nodes = {})

void **execute**()  
Executes this action.

void **execute**(*sh::actions::ActionExecutionInfo* \*info)  
Executes this action.

QKeySequence **shortcuthint**()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory**()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint**(QKeySequence shortcut, bool triggersOnCurrentDirectory = false)  
Sets the keyboard shortcut for triggering this action.

QString **text**()  
Returns the displayed text for this action.

QString **icon**()  
Returns the icon for this action.

bool **enabled**()  
Checks if this action is enabled.

bool **isChecked**()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable**()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () const

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>

> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **changed** ()

Emits when some data changed.

## Public Static Functions

void **doInitialize** ()

void **doShutdown** ()

**class ActionNumberFileviews\_Remove : public [sh::actions::ActionActionItem](#)**  
*#include <actionnumberfileviews.h>* Action for removing a certain fileview.

## Public Functions

**ActionNumberFileviews\_Remove** (int *i*, bool *deflt*)

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

```
void _setParentAction (std::shared_ptr<AbstractActionItem> parent)  
    Sets the parent action. .  
  
void initializeAsync (std::function<void>  
    > oninitialized = 0) Asynchronously initializes the action.  
  
void initializeSync ()  
    Synchronously initializes the action.  
  
bool isInitialized ()  
    Checks if this action is initialized.  
  
bool isInitializing ()  
    Checks if this action is initializing.
```

## Signals

```
void changed ()  
    Emits when some data changed.
```

```
class ActionNumberFileviews_Remove_Current : public sh::actions::ActionActionItem  
    #include <actionnumberfileviews.h> Action for removing the current fileview (for keyboard shortcut).
```

## Public Functions

```
ActionNumberFileviews_Remove_Current (QList<std::shared_ptr<sh::filesystem::FilesystemNode>>  
    nodes)  
  
void execute ()  
    Executes this action.  
  
void execute (sh::actions::ActionExecutionInfo *info)  
    Executes this action.  
  
QKeySequence shortcuthint ()  
    Returns the keyboard shortcut for triggering this action.  
  
bool shortcuthintTriggersOnCurrentDirectory ()  
    Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the  
    entry selection).  
  
void setShortcuthint (QKeySequence shortcut, bool triggersOnCurrentDirectory = false)  
    Sets the keyboard shortcut for triggering this action.  
  
QString text ()  
    Returns the displayed text for this action.  
  
QString icon ()  
    Returns the icon for this action.  
  
bool enabled ()  
    Checks if this action is enabled.  
  
bool isChecked ()  
    Checks if this action is checked (has a cross in the ui).  
  
bool isCheckable ()  
    Checks if this action is checkable (can have a cross in the ui).
```



int **defaultActionPrecedence** () const

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>

> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **changed** ()

Emits when some data changed.

## Public Static Functions

void **doInitialize** ()

void **doShutdown** ()

```
class ActionQuit : public sh::actions::ActionActionItem
#include <actionquit.h> Action for closing Shallot.
```

## Public Functions

**ActionQuit** ()

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo \*info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

```

void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.

```

## Signals

```

void changed ()
    Emits when some data changed.

```

```

class ActionRefresh : public sh::actions::ActionActionItem
    #include <actionrefresh.h> Action for refreshing the current view.

```

## Public Functions

```

ActionRefresh ()

void execute ()
    Executes this action.

void execute (sh::actions::ActionExecutionInfo *info)
    Executes this action.

QKeySequence shortcuthint ()
    Returns the keyboard shortcut for triggering this action.

bool shortcuthintTriggersOnCurrentDirectory ()
    Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the
    entry selection).

void setShortcuthint (QKeySequence shortcut, bool triggersOnCurrentDirectory = false)
    Sets the keyboard shortcut for triggering this action.

QString text ()
    Returns the displayed text for this action.

QString icon ()
    Returns the icon for this action.

bool enabled ()
    Checks if this action is enabled.

bool isChecked ()
    Checks if this action is checked (has a cross in the ui).

bool isCheckable ()
    Checks if this action is checkable (can have a cross in the ui).

int defaultActionPrecedence () const
    Returns the precedence for becoming the default action of a parent submenu.

```

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

```
void setText (QString text)  
    Sets the displayed text.  
  
void setIcon (QString icon)  
    Sets the icon.  
  
void setEnabled (bool enabled)  
    Sets if the item is enabled.  
  
void setChecked (bool checked)  
    Sets if the item is checked (has a cross in the ui).  
  
void setVisible (bool visible)  
    Sets the visibility of this item.  
  
bool visible ()  
    Checks the visibility of this item (non-recursively).  
  
std::weak_ptr<AbstractActionItem> parentAction ()  
    Returns the parent action, if it is added to a container.  
  
void _setParentAction (std::shared_ptr<AbstractActionItem> parent)  
    Sets the parent action. .  
  
void initializeAsync (std::function<void>  
    > oninitialized = 0) Asynchronously initializes the action.  
  
void initializeSync ()  
    Synchronously initializes the action.  
  
bool isInitialized ()  
    Checks if this action is initialized.  
  
bool isInitializing ()  
    Checks if this action is initializing.
```

## Signals

```
void changed ()  
    Emits when some data changed.
```

```
class ActionSaveSettings : public sh::actions::ActionActionItem  
    #include <actionsavesettings.h> Action for opening the ‘Save settings’ dialog.
```

## Public Functions

```
ActionSaveSettings ()  
  
void execute ()  
    Executes this action.  
  
void execute (sh::actions::ActionExecutionInfo *info)  
    Executes this action.  
  
QKeySequence shortcuthint ()  
    Returns the keyboard shortcut for triggering this action.
```

bool **shortcutHintTriggersOnCurrentDirectory** ()  
 Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcutHint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
 Sets the keyboard shortcut for triggering this action.

QString **text** ()  
 Returns the displayed text for this action.

QString **icon** ()  
 Returns the icon for this action.

bool **enabled** ()  
 Checks if this action is enabled.

bool **isChecked** ()  
 Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
 Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
 Returns the precedence for becoming the default action of a parent submenu.  
 This e.g. leads to a bold label.  
 The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
 Sets the displayed text.

void **setIcon** (QString *icon*)  
 Sets the icon.

void **setEnabled** (bool *enabled*)  
 Sets if the item is enabled.

void **setChecked** (bool *checked*)  
 Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
 Sets the visibility of this item.

bool **visible** ()  
 Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
 Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
 Sets the parent action. .

void **initializeAsync** (std::function<void>  
 > *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
 Synchronously initializes the action.

bool **isInitialized** ()  
 Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

**class ActionSearch** : public *sh::actions::ActionActionItem*  
*#include <actionsearch.h>* Action for opening the 'Save settings' dialog.

## Public Functions

**ActionSearch** ()

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo \*info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [\*ActionDefaultPrecedenceValues\*](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

```

void setEnabled (bool enabled)
    Sets if the item is enabled.

void setChecked (bool checked)
    Sets if the item is checked (has a cross in the ui).

void setVisible (bool visible)
    Sets the visibility of this item.

bool visible ()
    Checks the visibility of this item (non-recursively).

std::weak_ptr<AbstractActionItem> parentAction ()
    Returns the parent action, if it is added to a container.

void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.

```

## Signals

```

void changed ()
    Emits when some data changed.

```

```

class ActionSelectAllNodes : public sh::actions::mainmenu::ActionAbstractSelectNodes
    #include <actionselect.h> Action for selecting all nodes in the current file view.

```

## Public Functions

```

ActionSelectAllNodes ()

void execute ()
    Executes this action.

void execute (sh::actions::ActionExecutionInfo *info)
    Executes this action.

QKeySequence shortcuthint ()
    Returns the keyboard shortcut for triggering this action.

bool shortcuthintTriggersOnCurrentDirectory ()
    Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the
    entry selection).

void setShortcuthint (QKeySequence shortcut, bool triggersOnCurrentDirectory = false)
    Sets the keyboard shortcut for triggering this action.

QString text ()
    Returns the displayed text for this action.

```

QString **icon** ()

Returns the icon for this action.

bool **enabled** ()

Checks if this action is enabled.

bool **isChecked** ()

Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>

> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.



## Signals

void **changed** ()  
Emits when some data changed.

**class ActionSetWindowTitlePattern** : public *sh::actions::ActionActionItem*  
*#include <actionsetwindowtitlepattern.h>* Action for setting the window title pattern.

## Public Functions

**ActionSetWindowTitlePattern** ()

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \*info)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

**class ActionShowDetailPanel : public [sh::actions::ActionActionItem](#)**  
*#include <actionshowdetailpanel.h>* Action for toggling the visibility of the detail panel.

## Public Functions

**ActionShowDetailPanel** ()

void **execute** ()  
Executes this action.

void **execute** ([sh::actions::ActionExecutionInfo](#) \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
 Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
 Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
 Returns the precedence for becoming the default action of a parent submenu.  
 This e.g. leads to a bold label.  
 The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
 Sets the displayed text.

void **setIcon** (QString *icon*)  
 Sets the icon.

void **setEnabled** (bool *enabled*)  
 Sets if the item is enabled.

void **setChecked** (bool *checked*)  
 Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
 Sets the visibility of this item.

bool **visible** ()  
 Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
 Returns the parent action, if it is added to a container.

void **\_\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
 Sets the parent action. .

void **initializeAsync** (std::function<void>  
 > *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
 Synchronously initializes the action.

bool **isInitialized** ()  
 Checks if this action is initialized.

bool **isInitializing** ()  
 Checks if this action is initializing.

## Signals

void **changed** ()  
 Emits when some data changed.

**class ActionShowFolderTree**: public [sh::actions::ActionActionItem](#)  
*#include <actionshowfoldertree.h>* Action for toggling the visibility of the directory tree.

## Public Functions

**ActionShowFolderTree** ()

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo \*info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

```

void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.

```

## Signals

```

void changed ()
    Emits when some data changed.

```

```

class ActionShowLog : public sh::actions::ActionActionItem
    #include <actionshowlog.h> Action for showing the Shallot log in a dialog.

```

## Public Functions

```

ActionShowLog ()

void execute ()
    Executes this action.

void execute (sh::actions::ActionExecutionInfo *info)
    Executes this action.

QKeySequence shortcuthint ()
    Returns the keyboard shortcut for triggering this action.

bool shortcuthintTriggersOnCurrentDirectory ()
    Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the
    entry selection).

void setShortcuthint (QKeySequence shortcut, bool triggersOnCurrentDirectory = false)
    Sets the keyboard shortcut for triggering this action.

QString text ()
    Returns the displayed text for this action.

QString icon ()
    Returns the icon for this action.

bool enabled ()
    Checks if this action is enabled.

bool isChecked ()
    Checks if this action is checked (has a cross in the ui).

bool isCheckable ()
    Checks if this action is checkable (can have a cross in the ui).

int defaultActionPrecedence () const
    Returns the precedence for becoming the default action of a parent submenu.

```

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>)

> *oninitialized* = 0 Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **changed** ()

Emits when some data changed.

**class ActionTreeStickyness** : public [sh::actions::ActionActionItem](#)

*#include <actiontreestickyness.h>* Action for toggling the stickyness of the directory tree to the file-views.

## Public Functions

**ActionTreeStickyness** ()

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo \*info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

```
void _setParentAction (std::shared_ptr<AbstractActionItem> parent)  
    Sets the parent action. .  
  
void initializeAsync (std::function<void>  
    > oninitialized = 0) Asynchronously initializes the action.  
  
void initializeSync ()  
    Synchronously initializes the action.  
  
bool isInitialized ()  
    Checks if this action is initialized.  
  
bool isInitializing ()  
    Checks if this action is initializing.
```

## Signals

```
void changed ()  
    Emits when some data changed.
```

```
class ActionTuning : public sh::actions::ActionActionItem  
    #include <actiontuning.h> Action for opening the 'Tuning' dialog.
```

## Public Functions

```
ActionTuning ()
```

```
void execute ()  
    Executes this action.
```

```
void execute (sh::actions::ActionExecutionInfo *info)  
    Executes this action.
```

```
QKeySequence shortcuthint ()  
    Returns the keyboard shortcut for triggering this action.
```

```
bool shortcuthintTriggersOnCurrentDirectory ()  
    Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the  
    entry selection).
```

```
void setShortcuthint (QKeySequence shortcut, bool triggersOnCurrentDirectory = false)  
    Sets the keyboard shortcut for triggering this action.
```

```
QString text ()  
    Returns the displayed text for this action.
```

```
QString icon ()  
    Returns the icon for this action.
```

```
bool enabled ()  
    Checks if this action is enabled.
```

```
bool isChecked ()  
    Checks if this action is checked (has a cross in the ui).
```

```
bool isCheckable ()  
    Checks if this action is checkable (can have a cross in the ui).
```

```
int defaultActionPrecedence () const  
    Returns the precedence for becoming the default action of a parent submenu.
```



This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

```
void setText (QString text)
    Sets the displayed text.

void setIcon (QString icon)
    Sets the icon.

void setEnabled (bool enabled)
    Sets if the item is enabled.

void setChecked (bool checked)
    Sets if the item is checked (has a cross in the ui).

void setVisible (bool visible)
    Sets the visibility of this item.

bool visible ()
    Checks the visibility of this item (non-recursively).

std::weak_ptr<AbstractActionItem> parentAction ()
    Returns the parent action, if it is added to a container.

void __setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.
```

## Signals

```
void changed ()
    Emits when some data changed.
```

### 10.2.4 Namespace `sh::actions::common`

**namespace** `sh::actions::common`

Implementations of actions.

Subclasses of `sh::actions::AbstractActionItem` (and possibly some auxiliary stuff). Many action implementations used in Shallot are here. You can find other actions in the sibling namespaces and a few very special ones at different places.

**class** `ActionAbstractTransferTo` : **public** `sh::actions::SubmenuActionItem`

`#include <actiontransferto.h>` Abstract action for transferring a selection to some other locations (e.g. to other file views, bookmarks).

Subclassed by `sh::actions::common::ActionTransferToCopy`, `sh::actions::common::ActionTransferToMove`

## Public Functions

**ActionAbstractTransferTo** (QString *text*, QString *icon*,  
QList<std::shared\_ptr<sh::filesystem::FilesystemNode>>  
*nodes*)

**const** QList<std::shared\_ptr<sh::actions::AbstractActionItem>> **subitems** ()  
Returns the list of subitems from this submenu.

void **setSubitems** (**const** QList<std::shared\_ptr<sh::actions::AbstractActionItem>> *subitems*)  
Sets the subitems.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<AbstractActionItem> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<AbstractActionItem> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized**()  
Checks if this action is initialized.

bool **isInitializing**()  
Checks if this action is initializing.

## Signals

void **subitemsChanged**()  
Emits when the list of subitems changed.

void **changed**()  
Emits when some data changed.

## Friends

**friend class** ActionTransferToHelper

**class ActionAbstractTransferTree** : public *sh::actions::ActionActionItem*  
*#include <actionabstracttransfertree.h>* Abstract action for transferring (copying, moving, et al) a tree of items.

Subclassed by *sh::actions::common::ActionCopyTree*

## Public Functions

**ActionAbstractTransferTree** (QList<std::shared\_ptr<const *sh::filesystem::Eurl*>>  
*sources*, std::shared\_ptr<const *sh::filesystem::Eurl*>  
*destination*)

**~ActionAbstractTransferTree** ()

void **execute**()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckedable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () const

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>

> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **changed** ()

Emits when some data changed.

## Public Static Functions

void **makereadableunits** (double \*value, QString \*vunit)

**class MyFilesystemOperationProgressMonitor** : public *sh::filesystem::FilesystemOperationProgressMon*  
*#include <actionabstracttransfertree.h>*

## Public Functions

**MyFilesystemOperationProgressMonitor** (*ActionExecutionInfo* \*info)

bool **hasItemInfo** ()

Checks if this progress monitor provides information about how many items of a certain total number are transferred so far.

quint64 **doneItems** ()

Checks how many items are transferred so far.

quint64 **allItems** ()

Checks how many items are to be transferred in total (as predicted in the current moment).

bool **hasBytesInfo** ()

Checks if this progress monitor provides information about how many byte of a certain total number are transferred so far.

quint64 **doneBytes** ()

Checks how many bytes are transferred so far.

quint64 **allBytes** ()

Checks how many bytes are to be transferred in total (as predicted in the current moment).

QString **getItemInfoFrom** ()

Returns the current source of transfer (as textual information).

QString **getItemInfoTo** ()

Returns the current destination of transfer (as textual information).

QString **estimation** ()

Returns the current performance and time estimation (as textual information).

**class ActionAddToBookmarks** : public *sh::actions::ActionActionItem*  
*#include <actionbookmarks.h>* Action for bookmarking a directory.

## Public Functions

**ActionAddToBookmarks** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> nodes)

void **execute** ()

Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \*info)

Executes this action.

QKeySequence **shortcuthint** ()

Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()

Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcut** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

## Public Static Functions

void **doInitialize** ()

void **doShutdown** ()

**class ActionBookmark** : public *sh::actions::ActionActionItem*  
*#include <actionbookmarks.h>* Action for navigating to a certain bookmark.

## Public Functions

**ActionBookmark** (QString *label*, std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*)

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

```
void setIcon (QString icon)  
    Sets the icon.  
  
void setEnabled (bool enabled)  
    Sets if the item is enabled.  
  
void setChecked (bool checked)  
    Sets if the item is checked (has a cross in the ui).  
  
void setVisible (bool visible)  
    Sets the visibility of this item.  
  
bool visible ()  
    Checks the visibility of this item (non-recursively).  
  
std::weak_ptr<AbstractActionItem> parentAction ()  
    Returns the parent action, if it is added to a container.  
  
void __setParentAction (std::shared_ptr<AbstractActionItem> parent)  
    Sets the parent action. .  
  
void initializeAsync (std::function<void>  
    > oninitialized = 0) Asynchronously initializes the action.  
  
void initializeSync ()  
    Synchronously initializes the action.  
  
bool isInitialized ()  
    Checks if this action is initialized.  
  
bool isInitializing ()  
    Checks if this action is initializing.
```

## Signals

```
void changed ()  
    Emits when some data changed.
```

```
class ActionBookmarkFolder : public sh::actions::SubmenuActionItem  
    #include <actionbookmarks.h> Submenu action of bookmarks.  
  
    Subclassed by sh::actions::common::ActionBookmarks
```

## Public Functions

```
ActionBookmarkFolder (QString label)  
  
void addSubitem (std::shared_ptr<sh::actions::AbstractActionItem> item)  
  
void clearSubitems ()  
  
const QList<std::shared_ptr<sh::actions::AbstractActionItem>> subitems ()  
    Returns the list of subitems from this submenu.  
  
void setSubitems (const QList<std::shared_ptr<sh::actions::AbstractActionItem>> subitems)  
    Sets the subitems.  
  
QString text ()  
    Returns the displayed text for this action.  
  
QString icon ()  
    Returns the icon for this action.
```



bool **enabled** ()  
 Checks if this action is enabled.

bool **isChecked** ()  
 Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
 Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
 Returns the precedence for becoming the default action of a parent submenu.  
 This e.g. leads to a bold label.  
 The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
 Sets the displayed text.

void **setIcon** (QString *icon*)  
 Sets the icon.

void **setEnabled** (bool *enabled*)  
 Sets if the item is enabled.

void **setChecked** (bool *checked*)  
 Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
 Sets the visibility of this item.

bool **visible** ()  
 Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
 Returns the parent action, if it is added to a container.

void **\_\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
 Sets the parent action. .

void **initializeAsync** (std::function<void>  
 > *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
 Synchronously initializes the action.

bool **isInitialized** ()  
 Checks if this action is initialized.

bool **isInitializing** ()  
 Checks if this action is initializing.

## Signals

void **subitemsChanged** ()  
Emits when the list of subitems changed.

void **changed** ()  
Emits when some data changed.

**class ActionBookmarks : public [sh::actions::common::ActionBookmarkFolder](#)**  
*#include <actionbookmarks.h>* Bookmarking action (the main one you see in the toolbar)

## Public Functions

**ActionBookmarks** ()

void **initialize** ()  
Initialize the action. This should make the time-consuming parts, e.g. for determining a label or enabled state.

void **addSubitem** (std::shared\_ptr<[sh::actions::AbstractActionItem](#)> item)

void **clearSubitems** ()

**const** QList<std::shared\_ptr<[sh::actions::AbstractActionItem](#)>> **subitems** ()  
Returns the list of subitems from this submenu.

void **setSubitems** (**const** QList<std::shared\_ptr<[sh::actions::AbstractActionItem](#)>> subitems)  
Sets the subitems.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString text)  
Sets the displayed text.

void **setIcon** (QString icon)  
Sets the icon.

void **setEnabled** (bool enabled)  
Sets if the item is enabled.

void **setChecked** (bool checked)  
Sets if the item is checked (has a cross in the ui).

```

void setVisible (bool visible)
    Sets the visibility of this item.

bool visible ()
    Checks the visibility of this item (non-recursively).

std::weak_ptr<AbstractActionItem> parentAction ()
    Returns the parent action, if it is added to a container.

void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.

```

## Signals

```

void subitemsChanged ()
    Emits when the list of subitems changed.

void changed ()
    Emits when some data changed.

```

```

class ActionClipboardCopy : public sh::actions::ActionActionItem
    #include <actionclipboard.h> Action for copying to clipboard.

```

## Public Functions

```

ActionClipboardCopy (QList<std::shared_ptr<sh::filesystem::FileSystemNode>> nodes)

void execute ()
    Executes this action.

void execute (sh::actions::ActionExecutionInfo *info)
    Executes this action.

QKeySequence shortcuthint ()
    Returns the keyboard shortcut for triggering this action.

bool shortcuthintTriggersOnCurrentDirectory ()
    Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry
    selection).

void setShortcuthint (QKeySequence shortcut, bool triggersOnCurrentDirectory = false)
    Sets the keyboard shortcut for triggering this action.

QString text ()
    Returns the displayed text for this action.

QString icon ()
    Returns the icon for this action.

```

`bool enabled ()`  
Checks if this action is enabled.

`bool isChecked ()`  
Checks if this action is checked (has a cross in the ui).

`bool isCheckable ()`  
Checks if this action is checkable (can have a cross in the ui).

`int defaultActionPrecedence () const`  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

`void setText (QString text)`  
Sets the displayed text.

`void setIcon (QString icon)`  
Sets the icon.

`void setEnabled (bool enabled)`  
Sets if the item is enabled.

`void setChecked (bool checked)`  
Sets if the item is checked (has a cross in the ui).

`void setVisible (bool visible)`  
Sets the visibility of this item.

`bool visible ()`  
Checks the visibility of this item (non-recursively).

`std::weak_ptr<AbstractActionItem> parentAction ()`  
Returns the parent action, if it is added to a container.

`void _setParentAction (std::shared_ptr<AbstractActionItem> parent)`  
Sets the parent action. .

`void initializeAsync (std::function<void>  
> oninitialized = 0)`  
Asynchronously initializes the action.

`void initializeSync ()`  
Synchronously initializes the action.

`bool isInitialized ()`  
Checks if this action is initialized.

`bool isInitializing ()`  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

## Public Static Functions

void **doInitialize** ()

void **doShutdown** ()

**class ActionClipboardCut** : public *sh::actions::ActionActionItem*  
*#include <actionclipboard.h>* Action for cutting to clipboard.

## Public Functions

**ActionClipboardCut** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> *nodes*)

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

```
void setIcon (QString icon)  
    Sets the icon.  
  
void setEnabled (bool enabled)  
    Sets if the item is enabled.  
  
void setChecked (bool checked)  
    Sets if the item is checked (has a cross in the ui).  
  
void setVisible (bool visible)  
    Sets the visibility of this item.  
  
bool visible ()  
    Checks the visibility of this item (non-recursively).  
  
std::weak_ptr<AbstractActionItem> parentAction ()  
    Returns the parent action, if it is added to a container.  
  
void __setParentAction (std::shared_ptr<AbstractActionItem> parent)  
    Sets the parent action. .  
  
void initializeAsync (std::function<void>  
    > oninitialized = 0) Asynchronously initializes the action.  
  
void initializeSync ()  
    Synchronously initializes the action.  
  
bool isInitialized ()  
    Checks if this action is initialized.  
  
bool isInitializing ()  
    Checks if this action is initializing.
```

## Signals

```
void changed ()  
    Emits when some data changed.
```

## Public Static Functions

```
void doInitialize ()  
  
void doShutdown ()
```

```
class ActionClipboardPaste : public sh::actions::ActionActionItem  
    #include <actionclipboard.h> Action for pasting from clipboard.
```

## Public Functions

```
ActionClipboardPaste (QList<std::shared_ptr<sh::filesystem::FilesystemNode>> nodes)  
  
bool shortcutTriggersOnFolder ()  
  
void execute ()  
    Executes this action.  
  
void execute (sh::actions::ActionExecutionInfo *info)  
    Executes this action.
```

**QKeySequence shortcutHint ()**  
Returns the keyboard shortcut for triggering this action.

**bool shortcutHintTriggersOnCurrentDirectory ()**  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

**void setShortcutHint (QKeySequence shortcut, bool triggersOnCurrentDirectory = false)**  
Sets the keyboard shortcut for triggering this action.

**QString text ()**  
Returns the displayed text for this action.

**QString icon ()**  
Returns the icon for this action.

**bool enabled ()**  
Checks if this action is enabled.

**bool isChecked ()**  
Checks if this action is checked (has a cross in the ui).

**bool isCheckable ()**  
Checks if this action is checkable (can have a cross in the ui).

**int defaultActionPrecedence () const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

**void setText (QString text)**  
Sets the displayed text.

**void setIcon (QString icon)**  
Sets the icon.

**void setEnabled (bool enabled)**  
Sets if the item is enabled.

**void setChecked (bool checked)**  
Sets if the item is checked (has a cross in the ui).

**void setVisible (bool visible)**  
Sets the visibility of this item.

**bool visible ()**  
Checks the visibility of this item (non-recursively).

**std::weak\_ptr<AbstractActionItem> parentAction ()**  
Returns the parent action, if it is added to a container.

**void \_setParentAction (std::shared\_ptr<AbstractActionItem> parent)**  
Sets the parent action. .

**void initializeAsync (std::function<void>  
> oninitialized = 0)** Asynchronously initializes the action.

**void initializeSync ()**  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

## Public Static Functions

void **doInitialize** ()

void **doShutdown** ()

**class ActionClipboardPasteAsLink** : public *sh::actions::ActionActionItem*  
*#include <actionclipboard.h>* Action for pasting from clipboard as link.

## Public Functions

**ActionClipboardPasteAsLink** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>>  
*nodes*)

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.



The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

```
void setText (QString text)
    Sets the displayed text.

void setIcon (QString icon)
    Sets the icon.

void setEnabled (bool enabled)
    Sets if the item is enabled.

void setChecked (bool checked)
    Sets if the item is checked (has a cross in the ui).

void setVisible (bool visible)
    Sets the visibility of this item.

bool visible ()
    Checks the visibility of this item (non-recursively).

std::weak_ptr<AbstractActionItem> parentAction ()
    Returns the parent action, if it is added to a container.

void __setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.
```

## Signals

```
void changed ()
    Emits when some data changed.
```

## Public Static Functions

```
void doInitialize ()

void doShutdown ()
```

```
class ActionCopyTree : public sh::actions::common::ActionAbstractTransferTree
    #include <actioncopytree.h> Action copying a filesystem tree to some other place.

    Subclassed by sh::actions::common::ActionMoveTree
```

## Public Functions

**ActionCopyTree** (QList<std::shared\_ptr<const *sh::filesystem::Eurl*>> *sources*,  
std::shared\_ptr<const *sh::filesystem::Eurl*> *destination*)

void **action** (*sh::actions::ActionExecutionInfo* \**info*)  
The action implementation, i.e. what the actions should actually do.

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

```
std::weak_ptr<AbstractActionItem> parentAction ()
    Returns the parent action, if it is added to a container.

void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.
```

## Signals

```
void changed ()
    Emits when some data changed.
```

## Public Static Functions

```
void makereadableunits (double *value, QString *vunit)

class ActionCreateFile : public sh::actions::ActionActionItem
    #include <actioncreatefile.h> Action for creating a new file.
```

## Public Functions

```
ActionCreateFile (QList<std::shared_ptr<sh::filesystem::FilesystemNode>> nodes)

void execute ()
    Executes this action.

void execute (sh::actions::ActionExecutionInfo *info)
    Executes this action.

QKeySequence shortcuthint ()
    Returns the keyboard shortcut for triggering this action.

bool shortcuthintTriggersOnCurrentDirectory ()
    Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry
    selection).

void setShortcuthint (QKeySequence shortcut, bool triggersOnCurrentDirectory = false)
    Sets the keyboard shortcut for triggering this action.

QString text ()
    Returns the displayed text for this action.

QString icon ()
    Returns the icon for this action.

bool enabled ()
    Checks if this action is enabled.
```

bool **isChecked** ()

Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>

> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **changed** ()

Emits when some data changed.

## Public Static Functions

void **doInitialize** ()

void **doShutdown** ()

**class ActionCreateFolder** : public *sh::actions::ActionActionItem*  
*#include <actioncreatefolder.h>* Action for creating a new directory.

## Public Functions

**ActionCreateFolder** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> *nodes*)

bool **shortcutTriggersOnFolder** ()

void **execute** ()

Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)

Executes this action.

QKeySequence **shortcuthint** ()

Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()

Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)

Sets the keyboard shortcut for triggering this action.

QString **text** ()

Returns the displayed text for this action.

QString **icon** ()

Returns the icon for this action.

bool **enabled** ()

Checks if this action is enabled.

bool **isChecked** ()

Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<*AbstractActionItem*> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_\_setParentAction** (std::shared\_ptr<*AbstractActionItem*> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

## Public Static Functions

void **doInitialize** ()  
void **doShutdown** ()

**class ActionDeleteItems : public *sh::actions::ActionActionItem***  
*#include <actiondeleteitems.h>* Action deleting a list of filesystem items.

## Public Functions

**ActionDeleteItems** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> *nodes*)

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcut** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

## Public Static Functions

void **doInitialize** ()

void **doShutdown** ()

**class ActionExecute** : public *sh::actions::ActionActionItem*  
*#include <actionexecute.h>* Action executing a file (if executable).

## Public Functions

**ActionExecute** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> *nodes*)

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [\*ActionDefaultPrecedenceValues\*](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.



```

void setIcon (QString icon)
    Sets the icon.

void setEnabled (bool enabled)
    Sets if the item is enabled.

void setChecked (bool checked)
    Sets if the item is checked (has a cross in the ui).

void setVisible (bool visible)
    Sets the visibility of this item.

bool visible ()
    Checks the visibility of this item (non-recursively).

std::weak_ptr<AbstractActionItem> parentAction ()
    Returns the parent action, if it is added to a container.

void __setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.

```

## Signals

```

void changed ()
    Emits when some data changed.

```

## Public Static Functions

```

void doInitialize ()

void doShutdown ()

```

```

class ActionGroups : public sh::actions::SubmenuActionItem
    #include <actiongroups.h> Action for showing registered actions grouped by an
    sh::actions::ActionCategory.

```

## Public Functions

```

ActionGroups (std::shared_ptr<sh::actions::ActionCategory> category)

void setGroupActions (std::shared_ptr<sh::actions::ActionInstantiation> rootsituation,
    QList<std::shared_ptr<sh::actions::ActionInstantiation>> selects,
    QList<std::shared_ptr<sh::actions::ActionInstantiation>> directs)

const QList<std::shared_ptr<sh::actions::AbstractActionItem>> subitems ()
    Returns the list of subitems from this submenu.

```

void **setSubitems** (const QList<std::shared\_ptr<[sh::actions::AbstractActionItem](#)>> subitems)  
Sets the subitems.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () const  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString text)  
Sets the displayed text.

void **setIcon** (QString icon)  
Sets the icon.

void **setEnabled** (bool enabled)  
Sets if the item is enabled.

void **setChecked** (bool checked)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool visible)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> parent)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **subitemsChanged** ()  
Emits when the list of subitems changed.

void **changed** ()  
Emits when some data changed.

**class ActionHistoryNavigate** : public *sh::actions::SubmenuItem*  
*#include <actionnavigation.h>* Abstract action for navigating in the directory history stack (stepwise or randomly).  
 Subclassed by *sh::actions::common::ActionHistoryNavigateBackward*,  
*sh::actions::common::ActionHistoryNavigateForward*

## Public Functions

**ActionHistoryNavigate** (QString *text*, std::function<QList<*sh::tools::HistoryTracker*<std::shared\_ptr<const *sh::filesystem::Eurl*>>::HistoryEntry>>  
 > *\_getentriesfct*QString *icon*

const QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> **subitems** ()  
Returns the list of subitems from this submenu.

void **setSubitems** (const QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> *subitems*)  
Sets the subitems.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () const  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [\*ActionDefaultPrecedenceValues\*](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<*AbstractActionItem*> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<*AbstractActionItem*> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0)  
Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **subitemsChanged** ()  
Emits when the list of subitems changed.

void **changed** ()  
Emits when some data changed.

**class ActionHistoryNavigateBackward** : public *sh::actions::common::ActionHistoryNavigate*  
*#include <actionnavigation.h>* Action for navigating backward in the directory history stack.

## Public Functions

**ActionHistoryNavigateBackward** ()

**const** QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> **subitems** ()  
Returns the list of subitems from this submenu.

void **setSubitems** (**const** QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> *subitems*)  
Sets the subitems.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () const

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>

> *oninitialized* = 0) asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **subitemsChanged** ()

Emits when the list of subitems changed.

void **changed** ()

Emits when some data changed.

**class ActionHistoryNavigateForward** : public [sh::actions::common::ActionHistoryNavigate](#)  
[#include <actionnavigation.h>](#) Action for navigating forward in the directory history stack.

## Public Functions

**ActionHistoryNavigateForward()**

**const** QList<std::shared\_ptr<[\*sh::actions::AbstractActionItem\*](#)>> **subitems** ()

Returns the list of subitems from this submenu.

void **setSubitems** (**const** QList<std::shared\_ptr<[\*sh::actions::AbstractActionItem\*](#)>> *subitems*)

Sets the subitems.

QString **text** ()

Returns the displayed text for this action.

QString **icon** ()

Returns the icon for this action.

bool **enabled** ()

Checks if this action is enabled.

bool **isChecked** ()

Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [\*ActionDefaultPrecedenceValues\*](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[\*AbstractActionItem\*](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[\*AbstractActionItem\*](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>

> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized**()  
Checks if this action is initialized.

bool **isInitializing**()  
Checks if this action is initializing.

## Signals

void **subitemsChanged**()  
Emits when the list of subitems changed.

void **changed**()  
Emits when some data changed.

**class ActionManageBookmarks : public *sh::actions::ActionActionItem***  
*#include <actionbookmarks.h>* Action for managing bookmarks.

## Public Functions

**ActionManageBookmarks**()

void **execute**()  
Executes this action.

void **execute**(*sh::actions::ActionExecutionInfo \*info*)  
Executes this action.

QKeySequence **shortcuthint**()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory**()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint**(QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text**()  
Returns the displayed text for this action.

QString **icon**()  
Returns the icon for this action.

bool **enabled**()  
Checks if this action is enabled.

bool **isChecked**()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable**()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence**() **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See *ActionDefaultPrecedenceValues* for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<AbstractActionItem> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<AbstractActionItem> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0)  
Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

**class ActionMoveTree** : public *sh::actions::common::ActionCopyTree*  
*#include <actionmovetree.h>* Action moving a filesystem tree to some other place.

## Public Functions

**ActionMoveTree** (QList<std::shared\_ptr<const *sh::filesystem::Eurl*>> *sources*,  
std::shared\_ptr<const *sh::filesystem::Eurl*> *destination*)

void **action** (*sh::actions::ActionExecutionInfo* \**info*)  
The action implementation, i.e. what the actions should actually do.

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.



bool **shortcutHintTriggersOnCurrentDirectory** ()  
 Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcutHint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
 Sets the keyboard shortcut for triggering this action.

QString **text** ()  
 Returns the displayed text for this action.

QString **icon** ()  
 Returns the icon for this action.

bool **enabled** ()  
 Checks if this action is enabled.

bool **isChecked** ()  
 Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
 Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () const  
 Returns the precedence for becoming the default action of a parent submenu.  
 This e.g. leads to a bold label.  
 The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
 Sets the displayed text.

void **setIcon** (QString *icon*)  
 Sets the icon.

void **setEnabled** (bool *enabled*)  
 Sets if the item is enabled.

void **setChecked** (bool *checked*)  
 Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
 Sets the visibility of this item.

bool **visible** ()  
 Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
 Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
 Sets the parent action. .

void **initializeAsync** (std::function<void>  
 > *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
 Synchronously initializes the action.

bool **isInitialized** ()  
 Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

## Public Static Functions

void **makereadableunits** (double \*value, QString \*vunit)

**class ActionNavigateInHistory** : public [\*sh::actions::ActionActionItem\*](#)  
*#include <actionnavigation.h>* Action for navigating to one particular place in the directory history stack.

## Public Functions

**ActionNavigateInHistory** ([\*sh::ui::FileView\*](#) \*filelist, int idx, std::shared\_ptr<const [\*sh::filesystem::Eurl\*](#)> eurl, bool isdefault)

void **execute** ()  
Executes this action.

void **execute** ([\*sh::actions::ActionExecutionInfo\*](#) \*info)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence shortcut, bool triggersOnCurrentDirectory = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () const  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [\*ActionDefaultPrecedenceValues\*](#) for reference values.

```

void setText (QString text)
    Sets the displayed text.

void setIcon (QString icon)
    Sets the icon.

void setEnabled (bool enabled)
    Sets if the item is enabled.

void setChecked (bool checked)
    Sets if the item is checked (has a cross in the ui).

void setVisible (bool visible)
    Sets the visibility of this item.

bool visible ()
    Checks the visibility of this item (non-recursively).

std::weak_ptr<AbstractActionItem> parentAction ()
    Returns the parent action, if it is added to a container.

void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.

```

## Signals

```

void changed ()
    Emits when some data changed.

```

```

class ActionOpenArchive : public sh::actions::ActionActionItem
    #include <actionopenarchive.h> Action opening a share archive (by creating a root node for it).

    Subclassed      by      sh::filesystemhandlers::ArchiveFilesystemHandler::ActionOpenTarArchive,
    sh::filesystemhandlers::ArchiveFilesystemHandler::ActionOpenZipArchive

```

## Public Functions

```

ActionOpenArchive (QList<std::shared_ptr<sh::filesystem::FilesystemNode>> nodes)

void execute ()
    Executes this action.

void execute (sh::actions::ActionExecutionInfo *info)
    Executes this action.

QKeySequence shortcuthint ()
    Returns the keyboard shortcut for triggering this action.

```

bool **shortcutHintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcutHint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () const  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0)Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
 Checks if this action is initializing.

## Signals

void **changed** ()  
 Emits when some data changed.

**class ActionOpenDirectoryInNewWindow** : public *sh::actions::ActionActionItem*  
*#include <actionopendirectoryinnewwindow.h>* Action for opening a file with a automatically chosen program.

## Public Functions

**ActionOpenDirectoryInNewWindow** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>>  
*nodes*)

void **execute** ()  
 Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \*info)  
 Executes this action.

QKeySequence **shortcuthint** ()  
 Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
 Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
 Sets the keyboard shortcut for triggering this action.

QString **text** ()  
 Returns the displayed text for this action.

QString **icon** ()  
 Returns the icon for this action.

bool **enabled** ()  
 Checks if this action is enabled.

bool **isChecked** ()  
 Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
 Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
 Returns the precedence for becoming the default action of a parent submenu.  
 This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
 Sets the displayed text.

void **setIcon** (QString *icon*)  
 Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

## Public Static Functions

void **doInitialize** ()  
void **doShutdown** ()

**class ActionOpenFile** : public [sh::actions::ActionActionItem](#)  
*#include <actionopenfile.h>* Action for opening a file with a automatically chosen program.

## Public Functions

**ActionOpenFile** (QList<std::shared\_ptr<[sh::filesystem::FilesystemNode](#)>> *nodes*)

void **execute** ()  
Executes this action.

void **execute** ([sh::actions::ActionExecutionInfo](#) \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcutHintTriggersOnCurrentDirectory** ()  
 Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcutHint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
 Sets the keyboard shortcut for triggering this action.

QString **text** ()  
 Returns the displayed text for this action.

QString **icon** ()  
 Returns the icon for this action.

bool **enabled** ()  
 Checks if this action is enabled.

bool **isChecked** ()  
 Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
 Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
 Returns the precedence for becoming the default action of a parent submenu.  
 This e.g. leads to a bold label.  
 The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
 Sets the displayed text.

void **setIcon** (QString *icon*)  
 Sets the icon.

void **setEnabled** (bool *enabled*)  
 Sets if the item is enabled.

void **setChecked** (bool *checked*)  
 Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
 Sets the visibility of this item.

bool **visible** ()  
 Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
 Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
 Sets the parent action. .

void **initializeAsync** (std::function<void>  
 > *oninitialized* = 0)  
 Asynchronously initializes the action.

void **initializeSync** ()  
 Synchronously initializes the action.

bool **isInitialized** ()  
 Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

## Public Static Functions

void **doInitialize** ()  
void **doShutdown** ()

**class ActionOpenFileWith**: public *sh::actions::SubmenuItem*  
*#include <actionopenfilewith.h>* Submenu action for opening a file with a manually chosen program.

## Public Functions

**ActionOpenFileWith** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> *nodes*)

const QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> **subitems** ()  
Returns the list of subitems from this submenu.

void **setSubitems** (const QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> *subitems*)  
Sets the subitems.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () const  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [\*ActionDefaultPrecedenceValues\*](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.



```

void setChecked (bool checked)
    Sets if the item is checked (has a cross in the ui).

void setVisible (bool visible)
    Sets the visibility of this item.

bool visible ()
    Checks the visibility of this item (non-recursively).

std::weak_ptr<AbstractActionItem> parentAction ()
    Returns the parent action, if it is added to a container.

void __setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.

```

## Signals

```

void subitemsChanged ()
    Emits when the list of subitems changed.

void changed ()
    Emits when some data changed.

```

## Public Static Functions

```

void waitProgramClosedIfNeeded (sh::actions::ActionExecutionInfo *info,   QStringList
                                filelist)

void doInitialize ()

void doShutdown ()

```

## Friends

```

friend class ActionOpenFile

class __ActionOpenFileWithEntry : public sh::actions::ActionActionItem

```

## Public Functions

**\_ActionOpenFileWithEntry** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>>  
nodes, QString name, QString cmd, QStringList arguments,  
QStringList rememberformimetype)

void **action** (*sh::actions::ActionExecutionInfo* \*info)  
The action implementation, i.e. what the actions should actually do.

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \*info)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence shortcut, bool triggersOnCurrentDirectory = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString text)  
Sets the displayed text.

void **setIcon** (QString icon)  
Sets the icon.

void **setEnabled** (bool enabled)  
Sets if the item is enabled.

void **setChecked** (bool checked)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool visible)  
Sets the visibility of this item.

```

bool visible ()
    Checks the visibility of this item (non-recursively).

std::weak_ptr<AbstractActionItem> parentAction ()
    Returns the parent action, if it is added to a container.

void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.

```

### Public Members

```

QString _command
QStringList _commandArguments
QList<std::shared_ptr<sh::filesystem::FileSystemNode>> _nodes

```

### Signals

```

void changed ()
    Emits when some data changed.

```

```

class _ActionOpenFileWithUI : public sh::actions::ActionActionItem

```

### Public Functions

```

_ActionOpenFileWithUI (QList<std::shared_ptr<sh::filesystem::FileSystemNode>> nodes,
    QString mimetype)

void action (sh::actions::ActionExecutionInfo *info)
    The action implementation, i.e. what the actions should actually do.

void execute ()
    Executes this action.

void execute (sh::actions::ActionExecutionInfo *info)
    Executes this action.

QKeySequence shortcuthint ()
    Returns the keyboard shortcut for triggering this action.

bool shortcuthintTriggersOnCurrentDirectory ()
    Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the
    entry selection).

void setShortcuthint (QKeySequence shortcut, bool triggersOnCurrentDirectory = false)
    Sets the keyboard shortcut for triggering this action.

```

QString **text** ()

Returns the displayed text for this action.

QString **icon** ()

Returns the icon for this action.

bool **enabled** ()

Checks if this action is enabled.

bool **isChecked** ()

Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>

> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Public Members

QList<std::shared\_ptr<*sh::filesystem::FileSystemNode*>> **\_nodes**

QString **\_mimetype**

## Signals

void **changed** ()

Emits when some data changed.

**class ActionOpenSharcArchive : public *sh::actions::ActionActionItem***  
*#include <actionopensharcarchive.h>* Action opening a sharc archive (by creating a root node for it).

## Public Functions

**ActionOpenSharcArchive** (QList<std::shared\_ptr<*sh::filesystem::FileSystemNode*>> *nodes*)

void **execute** ()

Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)

Executes this action.

QKeySequence **shortcuthint** ()

Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()

Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)

Sets the keyboard shortcut for triggering this action.

QString **text** ()

Returns the displayed text for this action.

QString **icon** ()

Returns the icon for this action.

bool **enabled** ()

Checks if this action is enabled.

bool **isChecked** ()

Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<AbstractActionItem> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_\_setParentAction** (std::shared\_ptr<AbstractActionItem> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

## Public Static Functions

void **doInitialize** ()  
void **doShutdown** ()

**class ActionOpenTerminal** : public *sh::actions::SubmenuActionItem*  
*#include <actionopenterminal.h>* Action submenu with actions for opening commandline terminals.

## Public Functions

**ActionOpenTerminal** (QString *dir*)

void **initialize** ()  
Initialize the action. This should make the time-consuming parts, e.g. for determining a label or enabled state.

**const** QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> **subitems** ()  
Returns the list of subitems from this submenu.

void **setSubitems** (const QList<std::shared\_ptr<[sh::actions::AbstractActionItem](#)>> subitems)  
Sets the subitems.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () const  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString text)  
Sets the displayed text.

void **setIcon** (QString icon)  
Sets the icon.

void **setEnabled** (bool enabled)  
Sets if the item is enabled.

void **setChecked** (bool checked)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool visible)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> parent)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> oninitialized = 0) asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **subitemsChanged** ()  
Emits when the list of subitems changed.

void **changed** ()  
Emits when some data changed.

**class ActionOpenTerminalAsRoot** : public *sh::actions::common::ActionOpenTerminalAsUser*  
*#include <actionopenterminal.h>* Action for opening a commandline terminal as root.

## Public Functions

**ActionOpenTerminalAsRoot** (QString *dir*)

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [\*ActionDefaultPrecedenceValues\*](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.



```

void setChecked (bool checked)
    Sets if the item is checked (has a cross in the ui).

void setVisible (bool visible)
    Sets the visibility of this item.

bool visible ()
    Checks the visibility of this item (non-recursively).

std::weak_ptr<AbstractActionItem> parentAction ()
    Returns the parent action, if it is added to a container.

void __setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.

```

## Signals

```

void changed ()
    Emits when some data changed.

```

```

class ActionOpenTerminalAsUser : public sh::actions::ActionActionItem
    #include <actionopenterminal.h> Action for opening a commandline terminal (with current user).

    Subclassed by sh::actions::common::ActionOpenTerminalAsRoot

```

## Public Functions

```

ActionOpenTerminalAsUser (QString dir)

void execute ()
    Executes this action.

void execute (sh::actions::ActionExecutionInfo *info)
    Executes this action.

QKeySequence shortcuthint ()
    Returns the keyboard shortcut for triggering this action.

bool shortcuthintTriggersOnCurrentDirectory ()
    Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry
    selection).

void setShortcuthint (QKeySequence shortcut, bool triggersOnCurrentDirectory = false)
    Sets the keyboard shortcut for triggering this action.

QString text ()
    Returns the displayed text for this action.

```

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

**class ActionRenameItem**: public *sh::actions::ActionActionItem*  
*#include <actionrenameitem.h>* Action for renaming items.

## Public Functions

**ActionRenameItem** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> *nodes*)

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

## Public Static Functions

void **doInitialize** ()  
void **doShutdown** ()

**class ActionShowProperties** : public [sh::actions::ActionActionItem](#)  
*#include <actionshowproperties.h>* Action for showing the properties dialog for filesystem nodes.

## Public Functions

**ActionShowProperties** (QList<std::shared\_ptr<[sh::filesystem::FilesystemNode](#)>> *nodes*)

void **execute** ()  
Executes this action.

void **execute** ([sh::actions::ActionExecutionInfo](#) \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

**QString text ()**  
Returns the displayed text for this action.

**QString icon ()**  
Returns the icon for this action.

**bool enabled ()**  
Checks if this action is enabled.

**bool isChecked ()**  
Checks if this action is checked (has a cross in the ui).

**bool isCheckable ()**  
Checks if this action is checkable (can have a cross in the ui).

**int defaultActionPrecedence () const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

**void setText (QString text)**  
Sets the displayed text.

**void setIcon (QString icon)**  
Sets the icon.

**void setEnabled (bool enabled)**  
Sets if the item is enabled.

**void setChecked (bool checked)**  
Sets if the item is checked (has a cross in the ui).

**void setVisible (bool visible)**  
Sets the visibility of this item.

**bool visible ()**  
Checks the visibility of this item (non-recursively).

**std::weak\_ptr<AbstractActionItem> parentAction ()**  
Returns the parent action, if it is added to a container.

**void \_setParentAction (std::shared\_ptr<AbstractActionItem> parent)**  
Sets the parent action. .

**void initializeAsync (std::function<void>  
> oninitialized = 0)**Asynchronously initializes the action.

**void initializeSync ()**  
Synchronously initializes the action.

**bool isInitialized ()**  
Checks if this action is initialized.

**bool isInitializing ()**  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

## Public Static Functions

void **doInitialize** ()  
void **doShutdown** ()

**class ActionTransferToCopy** : public *sh::actions::common::ActionAbstractTransferTo*  
*#include <actiontransferto.h>* Action for copying a selection to some other locations.

## Public Functions

**ActionTransferToCopy** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> *nodes*)

**const** QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> **subitems** ()  
Returns the list of subitems from this submenu.

void **setSubitems** (**const** QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> *subitems*)  
Sets the subitems.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [\*ActionDefaultPrecedenceValues\*](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

```

bool visible ()
    Checks the visibility of this item (non-recursively).

std::weak_ptr<AbstractActionItem> parentAction ()
    Returns the parent action, if it is added to a container.

void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.

```

## Signals

```

void subitemsChanged ()
    Emits when the list of subitems changed.

void changed ()
    Emits when some data changed.

```

## Public Static Functions

```

void doInitialize ()
    Only called by the Shallot infrastructure for initialization.

void doShutdown ()
    Only called by the Shallot infrastructure for initialization.

```

```

class ActionTransferToHelper : public sh::actions::ActionActionItem
    #include <actiontransferto.h> This action is used as subitems in ActionAbstractTransferTo.

```

## Public Functions

```

ActionTransferToHelper (QString text,    QString icon,    std::shared_ptr<const
    sh::filesystem::Eurl> transferto, std::shared_ptr<ActionAbstractTransferTo>
    root)

void execute ()
    Executes this action.

void execute (sh::actions::ActionExecutionInfo *info)
    Executes this action.

QKeySequence shortcuthint ()
    Returns the keyboard shortcut for triggering this action.

bool shortcuthintTriggersOnCurrentDirectory ()
    Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry
    selection).

```

void **setShortcut** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.



## Signals

void **changed** ()  
Emits when some data changed.

**class ActionTransferToMove** : public *sh::actions::common::ActionAbstractTransferTo*  
*#include <actiontransferto.h>* Action for moving a selection to some other locations.

## Public Functions

**ActionTransferToMove** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> *nodes*)

**const** QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> **subitems** ()  
Returns the list of subitems from this submenu.

void **setSubitems** (**const** QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> *subitems*)  
Sets the subitems.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See *ActionDefaultPrecedenceValues* for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<*AbstractActionItem*> **parentAction** ()  
Returns the parent action, if it is added to a container.

```
void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.
```

### Signals

```
void subitemsChanged ()
    Emits when the list of subitems changed.

void changed ()
    Emits when some data changed.
```

### Public Static Functions

```
void doInitialize ()

void doShutdown ()
```

## 10.2.5 Namespace `sh::actions::mainmenu`

**namespace** `sh::actions::mainmenu`

Implementation of the main menu.

Subclasses of `sh::actions::AbstractActionItem` (and possibly some auxiliary stuff). You can find other actions in the sibling namespaces and a few very special ones at different places.

```
class ActionAbout : public sh::actions::ActionActionItem
    #include <actionabout.h> Action showing the Shallot about box.
```

### Public Functions

```
ActionAbout ()

void execute ()
    Executes this action.

void execute (sh::actions::ActionExecutionInfo *info)
    Executes this action.

QKeySequence shortcuthint ()
    Returns the keyboard shortcut for triggering this action.

bool shortcuthintTriggersOnCurrentDirectory ()
    Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).
```

void **setShortcut** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
 Sets the keyboard shortcut for triggering this action.

QString **text** ()  
 Returns the displayed text for this action.

QString **icon** ()  
 Returns the icon for this action.

bool **enabled** ()  
 Checks if this action is enabled.

bool **isChecked** ()  
 Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
 Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
 Returns the precedence for becoming the default action of a parent submenu.  
 This e.g. leads to a bold label.  
 The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
 Sets the displayed text.

void **setIcon** (QString *icon*)  
 Sets the icon.

void **setEnabled** (bool *enabled*)  
 Sets if the item is enabled.

void **setChecked** (bool *checked*)  
 Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
 Sets the visibility of this item.

bool **visible** ()  
 Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
 Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
 Sets the parent action. .

void **initializeAsync** (std::function<void>  
 > *oninitialized* = 0) asynchronously initializes the action.

void **initializeSync** ()  
 Synchronously initializes the action.

bool **isInitialized** ()  
 Checks if this action is initialized.

bool **isInitializing** ()  
 Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

**class ActionAbstractSelectNodes : public *sh::actions::ActionActionItem***  
*#include <actionselect.h>* Abstract subclass for actions which select some nodes in the current file view.  
Subclassed by *sh::actions::mainmenu::ActionInvertNodeSelection*, *sh::actions::mainmenu::ActionSelectAllNodes*

## Public Functions

**ActionAbstractSelectNodes** (QString *text*)

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.  
The action with the highest value becomes the default. Only values >0 will be considered. See [\*ActionDefaultPrecedenceValues\*](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

```

void setChecked (bool checked)
    Sets if the item is checked (has a cross in the ui).

void setVisible (bool visible)
    Sets the visibility of this item.

bool visible ()
    Checks the visibility of this item (non-recursively).

std::weak_ptr<AbstractActionItem> parentAction ()
    Returns the parent action, if it is added to a container.

void __setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.

```

## Signals

```

void changed ()
    Emits when some data changed.

```

```

class ActionApplyProfile : public sh::actions::SubmenuItem
    #include <actionapplyprofile.h> Submenu action for switching the current profile.

```

## Public Functions

```

ActionApplyProfile ()

const QList<std::shared_ptr<sh::actions::AbstractActionItem>> subitems ()
    Returns the list of subitems from this submenu.

void setSubitems (const QList<std::shared_ptr<sh::actions::AbstractActionItem>> subitems)
    Sets the subitems.

QString text ()
    Returns the displayed text for this action.

QString icon ()
    Returns the icon for this action.

bool enabled ()
    Checks if this action is enabled.

bool isChecked ()
    Checks if this action is checked (has a cross in the ui).

bool isCheckable ()
    Checks if this action is checkable (can have a cross in the ui).

```

int **defaultActionPrecedence** () const

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>)

> *oninitialized* = 0 Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **subitemsChanged** ()

Emits when the list of subitems changed.

void **changed** ()

Emits when some data changed.

**class ActionApplyProfileX: public [sh::actions::ActionActionItem](#)**

Action for switching the current profile to a certain one.

## Public Functions

**ActionApplyProfileX** (QString *profilename*, bool *checked*)

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<*AbstractActionItem*> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>

> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **changed** ()

Emits when some data changed.

**class ActionFileViewChangeIconDimension** : public *sh::actions::ActionActionItem*

*#include <actionfileviewchangeicondimension.h>* Action for changing the icon size.

Subclassed by *sh::actions::mainmenu::ActionFileViewDecreaseIconDimension*,  
*sh::actions::mainmenu::ActionFileViewIncreaseIconDimension*

## Public Functions

**ActionFileViewChangeIconDimension** (QString *name*, int *direction*, QString *icon*, QKey-  
Sequence *shortcut*)

void **execute** ()

Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)

Executes this action.

QKeySequence **shortcuthint** ()

Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()

Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)

Sets the keyboard shortcut for triggering this action.

QString **text** ()

Returns the displayed text for this action.

QString **icon** ()

Returns the icon for this action.

bool **enabled** ()

Checks if this action is enabled.

bool **isChecked** ()

Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()

Checks if this action is checkable (can have a cross in the ui).



int **defaultActionPrecedence** () const

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>)

> *oninitialized* = 0 Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **changed** ()

Emits when some data changed.

**class ActionFileViewDecreaseIconDimension : public [sh::actions::mainmenu::ActionFileViewChangeIconDim](#)**  
**#include <actionfileviewchangeicondimension.h>** Action for decreasing the icon size.

## Public Functions

**ActionFileViewDecreaseIconDimension** ()

void **execute** ()

Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo \*info*)

Executes this action.

QKeySequence **shortcuthint** ()

Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()

Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)

Sets the keyboard shortcut for triggering this action.

QString **text** ()

Returns the displayed text for this action.

QString **icon** ()

Returns the icon for this action.

bool **enabled** ()

Checks if this action is enabled.

bool **isChecked** ()

Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

```

void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.

```

## Signals

```

void changed ()
    Emits when some data changed.

```

```

class ActionFileviewFilesizeFormatting : public sh::actions::ActionActionItem
    #include <actionfileviewfilesizeformatting.h> Action for toggling the filesize formatting mode for the current fileview.

```

## Public Functions

```

ActionFileviewFilesizeFormatting ()

void execute ()
    Executes this action.

void execute (sh::actions::ActionExecutionInfo *info)
    Executes this action.

QKeySequence shortcuthint ()
    Returns the keyboard shortcut for triggering this action.

bool shortcuthintTriggersOnCurrentDirectory ()
    Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void setShortcuthint (QKeySequence shortcut, bool triggersOnCurrentDirectory = false)
    Sets the keyboard shortcut for triggering this action.

QString text ()
    Returns the displayed text for this action.

QString icon ()
    Returns the icon for this action.

bool enabled ()
    Checks if this action is enabled.

bool isChecked ()
    Checks if this action is checked (has a cross in the ui).

bool isCheckable ()
    Checks if this action is checkable (can have a cross in the ui).

```

int **defaultActionPrecedence** () const

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>)

> *oninitialized* = 0 Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **changed** ()

Emits when some data changed.

**class ActionFileViewIncreaseIconDimension : public [sh::actions::mainmenu::ActionFileViewChangeIconDim](#)**

*#include <actionfileviewchangeicondimension.h>* Action for increasing the icon size.

## Public Functions

**ActionFileViewIncreaseIconDimension** ()

void **execute** ()

Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo \*info*)

Executes this action.

QKeySequence **shortcuthint** ()

Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()

Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)

Sets the keyboard shortcut for triggering this action.

QString **text** ()

Returns the displayed text for this action.

QString **icon** ()

Returns the icon for this action.

bool **enabled** ()

Checks if this action is enabled.

bool **isChecked** ()

Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<*AbstractActionItem*> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

**class ActionFileviewShowHiddenFiles : public *sh::actions::ActionActionItem***  
*#include <actionfileviewshowhiddenfiles.h>* Action which toggles the visibility of hidden files in the current fileview.

## Public Functions

**ActionFileviewShowHiddenFiles** ()

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () const

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>)

> *oninitialized* = 0 Asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **changed** ()

Emits when some data changed.

**class ActionFileviewView: public [sh::actions::ActionActionItem](#)**

*#include <actionfileviewview.h>* Action which toggles between icon- and listmode for the current fileview.

## Public Functions

**ActionFileviewView** ()

void **execute** ()

Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo \*info*)

Executes this action.

QKeySequence **shortcuthint** ()

Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()

Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)

Sets the keyboard shortcut for triggering this action.

QString **text** ()

Returns the displayed text for this action.

QString **icon** ()

Returns the icon for this action.

bool **enabled** ()

Checks if this action is enabled.

bool **isChecked** ()

Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.



void **\_setParentAction** (std::shared\_ptr<*AbstractActionItem*> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

**class ActionGotoDirectory : public *sh::actions::ActionActionItem***  
*#include <actiongotodirectory.h>* Action which toggles the visibility of hidden files in the current fileview.

## Public Functions

**ActionGotoDirectory** ()

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

```
void setText (QString text)  
    Sets the displayed text.  
  
void setIcon (QString icon)  
    Sets the icon.  
  
void setEnabled (bool enabled)  
    Sets if the item is enabled.  
  
void setChecked (bool checked)  
    Sets if the item is checked (has a cross in the ui).  
  
void setVisible (bool visible)  
    Sets the visibility of this item.  
  
bool visible ()  
    Checks the visibility of this item (non-recursively).  
  
std::weak_ptr<AbstractActionItem> parentAction ()  
    Returns the parent action, if it is added to a container.  
  
void _setParentAction (std::shared_ptr<AbstractActionItem> parent)  
    Sets the parent action. .  
  
void initializeAsync (std::function<void>  
    > oninitialized = 0) Asynchronously initializes the action.  
  
void initializeSync ()  
    Synchronously initializes the action.  
  
bool isInitialized ()  
    Checks if this action is initialized.  
  
bool isInitializing ()  
    Checks if this action is initializing.
```

## Signals

```
void changed ()  
    Emits when some data changed.
```

```
class ActionHelp : public sh::actions::ActionActionItem  
    #include <actionhelp.h> Action for opening the Shallot help system.
```

## Public Functions

```
ActionHelp ()  
  
void execute ()  
    Executes this action.  
  
void execute (sh::actions::ActionExecutionInfo *info)  
    Executes this action.  
  
QKeySequence shortcuthint ()  
    Returns the keyboard shortcut for triggering this action.
```

bool **shortcutHintTriggersOnCurrentDirectory** ()  
 Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcutHint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
 Sets the keyboard shortcut for triggering this action.

QString **text** ()  
 Returns the displayed text for this action.

QString **icon** ()  
 Returns the icon for this action.

bool **enabled** ()  
 Checks if this action is enabled.

bool **isChecked** ()  
 Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
 Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () const  
 Returns the precedence for becoming the default action of a parent submenu.  
 This e.g. leads to a bold label.  
 The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
 Sets the displayed text.

void **setIcon** (QString *icon*)  
 Sets the icon.

void **setEnabled** (bool *enabled*)  
 Sets if the item is enabled.

void **setChecked** (bool *checked*)  
 Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
 Sets the visibility of this item.

bool **visible** ()  
 Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
 Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
 Sets the parent action. .

void **initializeAsync** (std::function<void>  
 > *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
 Synchronously initializes the action.

bool **isInitialized** ()  
 Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

**class ActionInvertNodeSelection** : public *sh::actions::mainmenu::ActionAbstractSelectNodes*  
*#include <actionselect.h>* Action for inverting the node selection in the current file view.

## Public Functions

**ActionInvertNodeSelection** ()

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo \*info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [\*ActionDefaultPrecedenceValues\*](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

```

void setEnabled (bool enabled)
    Sets if the item is enabled.

void setChecked (bool checked)
    Sets if the item is checked (has a cross in the ui).

void setVisible (bool visible)
    Sets the visibility of this item.

bool visible ()
    Checks the visibility of this item (non-recursively).

std::weak_ptr<AbstractActionItem> parentAction ()
    Returns the parent action, if it is added to a container.

void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.

```

## Signals

```

void changed ()
    Emits when some data changed.

```

## Private Members

```

QList<std::shared_ptr<sh::filesystem::FilesystemNode>> wasselected

class ActionMain: public sh::actions::SubmenuItem, public sh::base::Singleton
    #include <actionmain.h> Submenu action providing the Shallot main menu.

```

## Public Functions

```

const QList<std::shared_ptr<sh::actions::AbstractActionItem>> subitems ()
    Returns the list of subitems from this submenu.

void setSubitems (const QList<std::shared_ptr<sh::actions::AbstractActionItem>> subitems)
    Sets the subitems.

QString text ()
    Returns the displayed text for this action.

QString icon ()
    Returns the icon for this action.

bool enabled ()
    Checks if this action is enabled.

```

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

void **doInitialize** ()  
Executes singleton initialization.

void **doShutdown** ()  
Executes singleton shutdown.

void **shutdown** ()  
Shutdown down this singleton.

bool **isAlive** ()  
Returns if this singleton is alive (true until its shutdown begins).

## Signals

void **subitemsChanged** ()  
Emits when the list of subitems changed.

void **changed** ()  
Emits when some data changed.

## Private Functions

**ActionMain** ()

**class ActionManageSavedSettings** : public *sh::actions::ActionActionItem*  
*#include <actionmanagesavedsettings.h>* Action for opening the ‘Manage saved settings’ dialog.

## Public Functions

**ActionManageSavedSettings** ()

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \*info)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<AbstractActionItem> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_\_setParentAction** (std::shared\_ptr<AbstractActionItem> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

**class ActionNumberFileviews : public sh::actions::SubmenuActionItem**  
*#include <actionnumberfileviews.h>* Submenu action for changing the number of fileviews.

## Public Functions

**ActionNumberFileviews** ()

**const** QList<std::shared\_ptr<sh::actions::AbstractActionItem>> **subitems** ()  
Returns the list of subitems from this submenu.

void **setSubitems** (**const** QList<std::shared\_ptr<sh::actions::AbstractActionItem>> *subitems*)  
Sets the subitems.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.



bool **isChecked** ()

Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>

> *oninitialized* = 0) asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **subitemsChanged** ()

Emits when the list of subitems changed.

void **changed** ()

Emits when some data changed.

## Friends

```
friend class ActionNumberFileviews_Add
friend class ActionNumberFileviews_Remove
friend class ActionNumberFileviews_Remove_Current

class ActionNumberFileviews_Add : public sh::actions::ActionActionItem
    #include <actionnumberfileviews.h> Action for adding a new fileview.
```

## Public Functions

```
ActionNumberFileviews_Add (QList<std::shared_ptr<sh::filesystem::FilesystemNode>>
                             nodes = { })

void execute ()
    Executes this action.

void execute (sh::actions::ActionExecutionInfo *info)
    Executes this action.

QKeySequence shortcuthint ()
    Returns the keyboard shortcut for triggering this action.

bool shortcuthintTriggersOnCurrentDirectory ()
    Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry
    selection).

void setShortcuthint (QKeySequence shortcut, bool triggersOnCurrentDirectory = false)
    Sets the keyboard shortcut for triggering this action.

QString text ()
    Returns the displayed text for this action.

QString icon ()
    Returns the icon for this action.

bool enabled ()
    Checks if this action is enabled.

bool isChecked ()
    Checks if this action is checked (has a cross in the ui).

bool isCheckable ()
    Checks if this action is checkable (can have a cross in the ui).

int defaultActionPrecedence () const
    Returns the precedence for becoming the default action of a parent submenu.

    This e.g. leads to a bold label.

    The action with the highest value becomes the default. Only values >0 will be considered. See
    ActionDefaultPrecedenceValues for reference values.

void setText (QString text)
    Sets the displayed text.

void setIcon (QString icon)
    Sets the icon.

void setEnabled (bool enabled)
    Sets if the item is enabled.
```

```

void setChecked (bool checked)
    Sets if the item is checked (has a cross in the ui).

void setVisible (bool visible)
    Sets the visibility of this item.

bool visible ()
    Checks the visibility of this item (non-recursively).

std::weak_ptr<AbstractActionItem> parentAction ()
    Returns the parent action, if it is added to a container.

void __setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.

```

## Signals

```

void changed ()
    Emits when some data changed.

```

## Public Static Functions

```

void doInitialize ()

void doShutdown ()

```

```

class ActionNumberFileviews_Remove : public sh::actions::ActionActionItem
    #include <actionnumberfileviews.h> Action for removing a certain fileview.

```

## Public Functions

```

ActionNumberFileviews_Remove (int i, bool deflt)

void execute ()
    Executes this action.

void execute (sh::actions::ActionExecutionInfo *info)
    Executes this action.

QKeySequence shortcuthint ()
    Returns the keyboard shortcut for triggering this action.

bool shortcuthintTriggersOnCurrentDirectory ()
    Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry
    selection).

```

void **setShortcut** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

**class ActionNumberFileviews\_Remove\_Current** : public [\*sh::actions::ActionActionItem\*](#)  
*#include <actionnumberfileviews.h>* Action for removing the current fileview (for keyboard shortcut).

## Public Functions

**ActionNumberFileviews\_Remove\_Current** (QList<std::shared\_ptr<[\*sh::filesystem::FilesystemNode\*](#)>>  
*nodes*)

void **execute** ()  
Executes this action.

void **execute** ([\*sh::actions::ActionExecutionInfo\*](#) \*info)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [\*ActionDefaultPrecedenceValues\*](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<*AbstractActionItem*> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_\_setParentAction** (std::shared\_ptr<*AbstractActionItem*> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

## Public Static Functions

void **doInitialize** ()  
void **doShutdown** ()

**class ActionQuit** : public *sh::actions::ActionActionItem*  
*#include <actionquit.h>* Action for closing Shallot.

## Public Functions

**ActionQuit** ()

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcutHint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
 Sets the keyboard shortcut for triggering this action.

QString **text** ()  
 Returns the displayed text for this action.

QString **icon** ()  
 Returns the icon for this action.

bool **enabled** ()  
 Checks if this action is enabled.

bool **isChecked** ()  
 Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
 Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
 Returns the precedence for becoming the default action of a parent submenu.  
 This e.g. leads to a bold label.  
 The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
 Sets the displayed text.

void **setIcon** (QString *icon*)  
 Sets the icon.

void **setEnabled** (bool *enabled*)  
 Sets if the item is enabled.

void **setChecked** (bool *checked*)  
 Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
 Sets the visibility of this item.

bool **visible** ()  
 Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
 Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
 Sets the parent action. .

void **initializeAsync** (std::function<void>  
 > *oninitialized* = 0) asynchronously initializes the action.

void **initializeSync** ()  
 Synchronously initializes the action.

bool **isInitialized** ()  
 Checks if this action is initialized.

bool **isInitializing** ()  
 Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

```
class ActionRefresh : public sh::actions::ActionActionItem  
#include <actionrefresh.h> Action for refreshing the current view.
```

## Public Functions

**ActionRefresh** ()

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \*info)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).



```

void setVisible (bool visible)
    Sets the visibility of this item.

bool visible ()
    Checks the visibility of this item (non-recursively).

std::weak_ptr<AbstractActionItem> parentAction ()
    Returns the parent action, if it is added to a container.

void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.

```

## Signals

```

void changed ()
    Emits when some data changed.

```

```

class ActionSaveSettings : public sh::actions::ActionActionItem
    #include <actionsavesettings.h> Action for opening the ‘Save settings’ dialog.

```

## Public Functions

```

ActionSaveSettings ()

void execute ()
    Executes this action.

void execute (sh::actions::ActionExecutionInfo *info)
    Executes this action.

QKeySequence shortcuthint ()
    Returns the keyboard shortcut for triggering this action.

bool shortcuthintTriggersOnCurrentDirectory ()
    Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry
    selection).

void setShortcuthint (QKeySequence shortcut, bool triggersOnCurrentDirectory = false)
    Sets the keyboard shortcut for triggering this action.

QString text ()
    Returns the displayed text for this action.

QString icon ()
    Returns the icon for this action.

bool enabled ()
    Checks if this action is enabled.

```

bool **isChecked** ()

Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>

> *oninitialized* = 0) asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **changed** ()

Emits when some data changed.

**class ActionSearch**: public [sh::actions::ActionActionItem](#)

*#include <actionsearch.h>* Action for opening the 'Save settings' dialog.

## Public Functions

### ActionSearch ()

void **execute** ()

Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo \*info*)

Executes this action.

QKeySequence **shortcuthint** ()

Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()

Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)

Sets the keyboard shortcut for triggering this action.

QString **text** ()

Returns the displayed text for this action.

QString **icon** ()

Returns the icon for this action.

bool **enabled** ()

Checks if this action is enabled.

bool **isChecked** ()

Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<*AbstractActionItem*> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

**class ActionSelectAllNodes : public *sh::actions::mainmenu::ActionAbstractSelectNodes***  
*#include <actionselect.h>* Action for selecting all nodes in the current file view.

## Public Functions

**ActionSelectAllNodes** ()

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

```
void setText (QString text)
    Sets the displayed text.

void setIcon (QString icon)
    Sets the icon.

void setEnabled (bool enabled)
    Sets if the item is enabled.

void setChecked (bool checked)
    Sets if the item is checked (has a cross in the ui).

void setVisible (bool visible)
    Sets the visibility of this item.

bool visible ()
    Checks the visibility of this item (non-recursively).

std::weak_ptr<AbstractActionItem> parentAction ()
    Returns the parent action, if it is added to a container.

void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.
```

## Signals

```
void changed ()
    Emits when some data changed.
```

```
class ActionSetWindowTitlePattern : public sh::actions::ActionActionItem
    #include <actionsetwindowtitlepattern.h> Action for setting the window title pattern.
```

## Public Functions

```
ActionSetWindowTitlePattern ()

void execute ()
    Executes this action.

void execute (sh::actions::ActionExecutionInfo *info)
    Executes this action.

QKeySequence shortcuthint ()
    Returns the keyboard shortcut for triggering this action.
```

bool **shortcutHintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcutHint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
 Checks if this action is initializing.

## Signals

void **changed** ()  
 Emits when some data changed.

**class ActionShowDetailPanel** : public *sh::actions::ActionActionItem*  
*#include <actionshowdetailpanel.h>* Action for toggling the visibility of the detail panel.

## Public Functions

**ActionShowDetailPanel** ()

void **execute** ()  
 Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo \*info*)  
 Executes this action.

QKeySequence **shortcuthint** ()  
 Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
 Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
 Sets the keyboard shortcut for triggering this action.

QString **text** ()  
 Returns the displayed text for this action.

QString **icon** ()  
 Returns the icon for this action.

bool **enabled** ()  
 Checks if this action is enabled.

bool **isChecked** ()  
 Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
 Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
 Returns the precedence for becoming the default action of a parent submenu.  
 This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See *ActionDefaultPrecedenceValues* for reference values.

void **setText** (QString *text*)  
 Sets the displayed text.

void **setIcon** (QString *icon*)  
 Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void> > *oninitialized* = 0)  
Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

**class ActionShowFolderTree** : public [sh::actions::ActionActionItem](#)  
*#include <actionshowfoldertree.h>* Action for toggling the visibility of the directory tree.

## Public Functions

**ActionShowFolderTree** ()

void **execute** ()  
Executes this action.

void **execute** ([sh::actions::ActionExecutionInfo](#) \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.



QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

**class ActionShowLog**: public *sh::actions::ActionActionItem*  
*#include <actionshowlog.h>* Action for showing the Shallot log in a dialog.

## Public Functions

**ActionShowLog** ()

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \*info)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

```

void setVisible (bool visible)
    Sets the visibility of this item.

bool visible ()
    Checks the visibility of this item (non-recursively).

std::weak_ptr<AbstractActionItem> parentAction ()
    Returns the parent action, if it is added to a container.

void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.

```

## Signals

```

void changed ()
    Emits when some data changed.

```

```

class ActionTreeStickyness : public sh::actions::ActionActionItem
    #include <actiontreestickyness.h> Action for toggling the stickyness of the directory tree to the fileviews.

```

## Public Functions

```

ActionTreeStickyness ()

void execute ()
    Executes this action.

void execute (sh::actions::ActionExecutionInfo *info)
    Executes this action.

QKeySequence shortcuthint ()
    Returns the keyboard shortcut for triggering this action.

bool shortcuthintTriggersOnCurrentDirectory ()
    Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry
    selection).

void setShortcuthint (QKeySequence shortcut, bool triggersOnCurrentDirectory = false)
    Sets the keyboard shortcut for triggering this action.

QString text ()
    Returns the displayed text for this action.

QString icon ()
    Returns the icon for this action.

bool enabled ()
    Checks if this action is enabled.

```

bool **isChecked** ()

Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<[AbstractActionItem](#)> *parent*)

Sets the parent action. .

void **initializeAsync** (std::function<void>

> *oninitialized* = 0) asynchronously initializes the action.

void **initializeSync** ()

Synchronously initializes the action.

bool **isInitialized** ()

Checks if this action is initialized.

bool **isInitializing** ()

Checks if this action is initializing.

## Signals

void **changed** ()

Emits when some data changed.

**class ActionTuning**: public [sh::actions::ActionActionItem](#)

*#include <actiontuning.h>* Action for opening the ‘Tuning’ dialog.

## Public Functions

### ActionTuning ()

void **execute** ()

Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo \*info*)

Executes this action.

QKeySequence **shortcutHint** ()

Returns the keyboard shortcut for triggering this action.

bool **shortcutHintTriggersOnCurrentDirectory** ()

Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcutHint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)

Sets the keyboard shortcut for triggering this action.

QString **text** ()

Returns the displayed text for this action.

QString **icon** ()

Returns the icon for this action.

bool **enabled** ()

Checks if this action is enabled.

bool **isChecked** ()

Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

std::weak\_ptr<[AbstractActionItem](#)> **parentAction** ()

Returns the parent action, if it is added to a container.

```
void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.
```

## Signals

```
void changed ()
    Emits when some data changed.
```

## 10.2.6 Namespace *sh::base*

**namespace** *sh::base*

Core logic.

Includes common data structures and some basic infrastructure. Some other namespaces in *sh* contain parts of the infrastructure as well.

## Enums

**enum** **LogSeverity**

The severity of a log message.

*Values:*

```
enumerator DEBUG = 0
enumerator INFO = 100
enumerator _DEFAULT = INFO
enumerator WARNING = 200
enumerator ERROR = 300
enumerator E_ERROR = ERROR
```

**class** **IconManager** : **public** **QObject**, **public** *sh::base::Singleton*

*#include <iconmanager.h>* Fetches icons according to theme availability and settings.

## Public Functions

**QIcon** **getIcon** (QString *mainname*, QString *emblemname* = QString(), QString *miniemblemname* = QString(), QStringList *tags* = QStringList())  
Creates a QIcon from the icon name(s). Uses a cache.

**QIcon** **getIconByFullname** (QString *fullname*)  
Creates a QIcon from the icon fullname. Uses a cache.

**QIcon** **getIcon** (QIcon *mainicon*, QString *emblemname*, QString *miniemblemname*, QStringList *tags*)  
Creates a QIcon based on an existing one. Uncached.

**QPixmap** **getPixmap** (QString *name*, int *size* = 22)  
Creates a QPixmap for an icon name. Uncached.

**~IconManager** ()

void **doInitialize** ()  
Executes singleton initialization.

void **doShutdown** ()  
Executes singleton shutdown.

void **shutdown** ()  
Shutdown down this singleton.

bool **isAlive** ()  
Returns if this singleton is alive (true until its shutdown begins).

## Private Functions

**IconManager** ()

**QIcon** **\_getIcon** (QString *name*)

**QImage** **\_colourImage** (QImage *img*, QColor *color*)

## Private Members

**QMutex** **cachemutex**

**QHash**<QString, QIcon> **cache**

**QList**<*GetIconStrategy*\*> **getIconStrategies**

std::shared\_ptr<*sh::configuration::ConfigurationValue*> **cfgvalPreferredStrategy**

**class** **GetIconStrategy**

Subclassed by *sh::base::IconManager::IncludedGetIconStrategy*,  
*sh::base::IconManager::QIconFromThemeGetIconStrategy*

### Public Functions

QIcon **getIcon** (QString *name*) = 0

~**GetIconStrategy** ()

**class** **IncludedGetIconStrategy** : **public** *sh::base::IconManager::GetIconStrategy*

### Public Functions

**IncludedGetIconStrategy** ()

QIcon **getIcon** (QString *name*)

### Private Members

QColor **brandingcolor1**

QColor **brandingcolor2**

**class** **QIconFromThemeGetIconStrategy** : **public** *sh::base::IconManager::GetIconStrategy*

### Public Functions

**QIconFromThemeGetIconStrategy** ()

QIcon **getIcon** (QString *name*)

### Private Members

QHash<QString, QString> **\_aliases**

**class** **Logger** : **public** QObject, **public** *sh::base::Singleton*

*#include <logger.h>* The logging manager.

Use it for writing messages to the Shallot log and for reading from it.

Note: Logging is easiest by the LOG\_\* macros like SH\_LOG\_INFO (and it provides more meta data!).

### Public Functions

void **logException** (*sh::exceptions::Exception* &*ex*, *LogSeverity* *severity* = *LogSeverity::ERROR*,  
                    QString *source* = QString())

Logs an exception.

void **log** (QString *message*, *LogSeverity* *severity*, QString *source* = QString())

Logs a message.

QString **getLogAsText** (*LogSeverity* *minseverity* = *LogSeverity::DEBUG*)

Returns all logged messages as one text.

QList<*LogMessage\**> **logMessages** ()

Returns the list of logged messages.

QString **logPrefix** ()

Returns the log prefix, i.e. an optional string all log output begins with.

~**Logger** ()



void **doInitialize** ()  
    Executes singleton initialization.

void **doShutdown** ()  
    Executes singleton shutdown.

void **shutdown** ()  
    Shutdown down this singleton.

bool **isAlive** ()  
    Returns if this singleton is alive (true until its shutdown begins).

## Signals

void **newLogMessageArrived** ()  
    Emitted when a new log message was written.

## Private Functions

**Logger** ()

## Private Members

QMutex **mutex**

QList<*LogMessage*\*> **\_logMessages**

QString **\_logPrefix**

**struct LogMessage**  
    *#include <logger.h>* A single log message.

## Public Members

QString **message**

*LogSeverity* **severity**

QString **source**

QDateTime **time**

**class MainThread**  
    *#include <mainthread.h>* The main thread.

Lots of data structure may only accessed from the main thread. The UI and the *sh::filesystem::FilesystemModel* also live in this thread (although some functions there may allow multi-threading in some ways).

## Public Static Functions

*sh::base::ThreadDispatcher* \***dispatcher** ()

The *sh::base::ThreadDispatcher*, which allows to execute code in the main thread.

**class ShallotProcess** : public *sh::base::Singleton*

*#include <shallotprocess.h>* Provides parameters given from the user by command line and some more simple infos.

## Public Functions

QString **uiMode** ()

The ui mode (e.g. “qt”, “web”) specified on command line.

QMap<QString, QString> **configurationValueAssignments** ()

The configuration value assignments set on command line.

QList<QString> **configurationValueFiles** ()

The configuration value files set on command line.

QString **parameterValue** (QString *key*, QString *deft* = QString())

Returns the command line parameter value for a key as string.

QStringList **parameterValueList** (QString *key*)

Returns the command line parameter value for a key as list of strings (for multiple parameter usage).

qint64 **parameterValueInt** (QString *key*, qint64 *deft* = 0)

Returns the command line parameter value for a key as integer.

QStringList **parameters** ()

Returns the list of keys of all specified command line parameter names.

QString **initialWorkDirectory** ()

Returns the initial directory specified on command line.

QString **logPrefix** ()

Returns the log prefix (i.e. an optional string all log output begins with) specified on command line.

QStringList **lang** ()

Returns the ui language specified on command line.

QColor **brandingColor** ()

Returns the Shallot branding color (typically a dark red).

int **minport** ()

Return the minimal allowed port specified on command line (for web ui).

int **maxport** ()

Return the maximal allowed port specified on command line (for web ui).

int **workerminport** ()

Return the minimal allowed port specified on command line for spawning web workers with (for web ui).

int **workermaxport** ()

Return the maximal allowed port specified on command line for spawning web workers with (for web ui).

int **maxnumberworkers** (int *deft*)

Maximum number of workers specified on command line.

int **maxnumberworkersperhost** (int *deflt*)  
Maximum number of workers per host specified on command line.

int **maxmemorypercent** (int *deflt*)  
Maximum memory in percent specified on command line.

int **maxmemorypercentglobal** (int *deflt*)  
Maximum memory in percent globally specified on command line.

int **maxbrowserawayseconds** (int *deflt*)  
Maximum browser away time in seconds specified on command line.

int **maxidlenessesseconds** (int *deflt*)  
Maximum user idleness time in seconds specified on command line.

void **doInitialize** ()  
Executes singleton initialization.

void **doShutdown** ()  
Executes singleton shutdown.

void **shutdown** ()  
Shutdown down this singleton.

bool **isAlive** ()  
Returns if this singleton is alive (true until its shutdown begins).

## Public Static Functions

QString **userDataDir** ()  
Returns the path to a storage for per-user data. It resides somewhere within the user home directory.

QString **shallotDataDir** ()  
Returns the directory of the Shallot program data. It contains the static files which are part of the Shallot program.

QString **revisionString** ()  
Returns the Shallot revision string, i.e. the exact version number. Note: This is only updated by a special build tool (anise) and not by directly compiling via qmake.

QString **homepage** ()  
Returns the url to the Shallot homepage.

QDateTime **buildtime** ()  
Returns the time when this Shallot build was created. Note: This is only updated by a special build tool (anise) and not by directly compiling via qmake.

## Private Functions

**ShallotProcess** ()

## Private Members

QMutex **\_brandingcolormutex**  
QMap<QString, QStringList> **\_cmdlineargs**  
QString **\_workdir**  
QString **\_ui**  
QMap<QString, QString> **\_cfgvalassignments**  
QList<QString> **\_cfgvalfiles**  
QColor **\_brandingcolor**  
std::shared\_ptr<*sh::configuration::ConfigurationValue*> **cfgvalBrandingColor**  
int **\_minport** = 0  
int **\_maxport** = 0  
int **\_workerminport** = 0  
int **\_workermaxport** = 0

## class Singleton

*#include <singletoninitializer.h>* A singleton with initialization on Shallot startup and shutdown in the end.

See *SingletonInitializer*.

Subclassed by *sh::actions::ActionsManager*, *sh::actions::mainmenu::ActionMain*,  
*sh::base::IconManager*, *sh::base::Logger*, *sh::base::ShallotProcess*,  
*sh::configuration::ConfigurationManager*, *sh::detailcolumns::DetailColumnCustomAttributes*,  
*sh::detailcolumns::DetailColumnDirectSymlinkTarget*, *sh::detailcolumns::DetailColumnExtendedAttributes*,  
*sh::detailcolumns::DetailColumnFilesize*, *sh::detailcolumns::DetailColumnMimetype*,  
*sh::detailcolumns::DetailColumnMtime*, *sh::detailcolumns::DetailColumnResolvedSymlink*,  
*sh::exceptions::ExceptionHandlerSettingsManager*, *sh::filesystem::FilesystemHandlerRegister*,  
*sh::filesystem::FilesystemModel*, *sh::filesystem::FilesystemModelDirectoryTreeProxy*,  
*sh::filesystem::FilesystemModelDirectoryTreeProxyVisibilityEnforcements*,  
*sh::filesystemhandlers::GnomeIODevicesFilesystemHandler*, *sh::filesystemhandlers::GnomeIONetworkFilesystemHandler*,  
*sh::filesystemhandlers::GnomeIOSmbFilesystemHandler*, *sh::filesystemhandlers::LocalFilesystemHandler*,  
*sh::filesystemhandlers::SharcFilesystemHandler*, *sh::paneldetails::PanelDetailManager*,  
*sh::scripting::api::ApiGlobalObject*, *sh::scripting::PythonScriptInterpreter*,  
*sh::scripting::ScriptingEngine*, *sh::search::SearchFilesystemHandler*, *sh::search::SearchManager*,  
*sh::settings::SettingsManager*, *sh::tools::accounts::AccountsManager*, *sh::tools::Benchmarking*,  
*sh::tools::BookmarkManager*, *sh::tools::DataExchange*, *sh::tools::filetypes::FileTypeManager*,  
*sh::tools::filetypes::UserDefinedOpenMethodDeterminationStrategy*, *sh::tools::LocalFilesystemWatcher*,  
*sh::tools::LocalFilesystemWatcherConnector*, *sh::tools::OperationsCache*,  
*sh::tools::ThumbnailManager*, *sh::tools::VisibleViews*

## Public Functions

```
void doInitialize ()
    Executes singleton initialization.

void doShutdown ()
    Executes singleton shutdown.

void shutdown ()
    Shutdown down this singleton.

bool isAlive ()
    Returns if this singleton is alive (true until its shutdown begins).

~Singleton ()

Singleton (const Singleton&) = delete
Singleton (Singleton&&) = delete
```

## Private Members

```
QMutex _mutex_shutdown
bool _shutdown = false
```

### class SingletonInitializer

*#include <singletoninitializer.h>* Takes care of initialization and shutdown of infrastructure singletons.

It is a very early part of shallot core infrastructure. Singletons typically participate by using those macros:

`DECLARE_SINGLETON(STYPE)`: Used inside the singleton class definition. `STYPE` is the singleton's class. It must be a subclass of `sh::base::Singleton`. It will get a static `std::shared_ptr<STYPE>` instance() method by that.

`REGISTER_SINGLETON(NS, STYPE, GROUPNAME)`: Used inside the source file. `NS` is the namespace of the singleton. `STYPE` is the singleton's class. `GROUPNAME` is the singleton group name (used for dependency handling, see later).

For singleton groups, there is `REGISTER_SINGLETON_GROUP(GROUPNAME, ...)`: `GROUPNAME` is the group name used for grouping singletons together. Additional arguments are other group names; those groups are considered as dependencies, which must be fulfilled before this group can.

For just initializing stuff via static methods (without a singleton instance), use `REGISTER_STATICINIT(NS, STYPE, GROUPNAME)`: `NS` is the namespace of the class to initialize. `STYPE` is the class to initialize. It must provide public static `void doInitialize()` and static `void doShutdown()`. `GROUPNAME` is the group name.

## Public Functions

```
void callInitializers ()
    Executes all create-callbacks at first, then all init-callbacks. In both runs, the callbacks with lower
    index come first. Afterwards, the singletons are considered as up and running.

SingletonInitializer ()
    Constructed only by the infrastructure and made available otherwise.

~SingletonInitializer ()
```

void **shutdown** ()

Executes all shutdown-callbacks at first, then all remove-callbacks. The callback order is the reversed one of callInitializers. Afterwards, the singletons are considered as shut down and removed.

### Public Static Functions

char **registerGroup** (QString *groupname*, QStringList *groupdeps*)

Registers a singleton group.

You should typically use REGISTER\_SINGLETON\_GROUP. See [SingletonInitializer](#).

char **registerSingleton** (QString *name*, std::function<std::shared\_ptr<[Singleton](#)>>()> *instancefct*, QString *groupname*) Registers a singleton.

You should typically use DECLARE\_SINGLETON and REGISTER\_SINGLETON. See [SingletonInitializer](#).

QThread \***initializerThread** ()

bool **isShutdown** ()

If the singletons are completely shut down and removed.

### Private Types

std::tuple< QString, QString, std::function< std::shared\_ptr< Singleton >>()> > > SingletonGroupTuple

typedef std::tuple<QString, QStringList> SingletonGroupTuple

### Private Static Functions

bool **dependsOn** (QString *group*, QString *depgroup*)

### Private Static Attributes

QList<SingletonTuple> \*\_**singletons**

QHash<QString, [SingletonGroupTuple](#)> \*\_**groups**

bool **\_isshutdown** = false

QMutex **shutdownmutex**

**class ThreadDispatcher** : public QObject

*#include <threaddispatcher.h>* Dispatcher for sync/async invocation of functions into the associated thread.

## Public Functions

**ThreadDispatcher** (QObject *\*parent*, QThread *\*thread*)

Constructed only by the infrastructure and made available otherwise.

void **invokeSync** (std::function<void>)

>Synchronously executes a function in the dispatcher's thread.

If it already runs in the current thread, it just executes the function. It returns when the function is completely executed.

void **invokeAsync** (std::function<void>)

>Asynchronously executes a function in the dispatcher's thread.

It directly returns. The actual execution will take place in a later message loop iteration of that thread.

void **invokeAsync** (std::function<void>)

>std::shared\_ptr<void> *obj1*, std::shared\_ptr<void> *obj2* = 0, std::shared\_ptr<void> *obj3* = 0Like the other variant, but it can also conserve some shared pointers until after execution.

bool **inThread** ()

Returns if the current thread is already equal to the dispatcher's thread (so dispatching isn't required).

## Signals

void **\_invoked** ()

## Private Functions

void **\_invokeAsync** (std::function<void>)  
>

## Private Members

QMutex **callsmutex**

QThread **\*\_thread**

QQueue<std::function<void ()>> **\_queue**

## Private Slots

void **slot\_invoked** ()

**class ThreadPool** : public QObject

*#include <threadpool.h>* A pool of worker threads doing some short jobs from a queue.

## Public Static Functions

void **enqueueForce** (std::function<void>  
> *fcn*) Enqueues code to the threadpool.

void **enqueueForce** (std::function<void>  
> *fcn*, std::shared\_ptr<void> *obj1*, std::shared\_ptr<void> *obj2* = 0, std::shared\_ptr<void> *obj3* = 0) En-  
queues code to the threadpool.

It can also conserve some shared pointers until after execution.

void **enqueueForceBeyondAsyncCallBarrier** (std::function<void>  
> *fcn*, std::shared\_ptr<void> *obj1* = 0, std::shared\_ptr<void> *obj2* = 0, std::shared\_ptr<void> *obj3* =  
0) Enqueues code to the threadpool and forbids async calls within it. This is useful for ensuring that no  
code might access a certain object after its deletion.

It can also conserve some shared pointers until after execution.

void **enqueueIfIn** (std::function<void>  
> *fcn*, QThread \**thread*) Enqueues code to the threadpool if currently in *thread*. Otherwise executes  
directly.

void **enqueueIfInMain** (std::function<void>  
> *fcn*) Enqueues code to the threadpool if currently in main thread. Otherwise executes directly.

void **doShutdown** ()  
Only called by the Shallot infrastructure for shutdown.

bool **isShuttingDown** ()

bool **isShutdown** ()

int **getThreadCount** ()

void **respectThreadAbort** ()

void **doInitialize** ()

## Private Static Functions

void **\_enqueueForce** (std::function<void>  
> *fcn*)

## Private Static Attributes

const int **THREAD\_COUNT** = 10

QMutex **poolmutex**

QWaitCondition **pooladdedcondition**

QList<std::function<void ()>> **tasks**

bool **\_shutdown** = false

QList<Thread\* > **\_threads**

int **\_threadsalive** = 0



## Friends

```
friend class ThreadPoolThread

class ThreadPoolThread : public QThread
#include <threadpool.h> A thread in the thread pool.
```

## Friends

```
friend class ThreadPool
```

## 10.2.7 Namespace sh::configuration

```
namespace sh::configuration
```

Configuration, so everything you see in ‘Finetuning’, and some other data (e.g. get shallot program directory).

## Enums

```
enum ConfigurationCategory
```

Categories of Configursh::configuration::ConfigurationValuentations. They are used for grouping them in the dialogs. There is no difference in behavior implied by this choice.

*Values:*

```
enumerator CategoryNone = 0
```

No category.

```
enumerator CategoryGUI = 1
```

User interface category.

```
enumerator CategoryBehavior = 2
```

Behavioral configuration category.

```
enumerator CategoryExternalTools = 3
```

External tools category.

```
class ConfigurationManager : public QObject, public sh::base::Singleton
```

```
#include <configurationmanager.h> Manages the Shallot configuration.
```

This is the more static part of Shallot. Much more stuff, i.e. everything you see in ‘Manage saved settings’, is in *sh::settings::SettingsManager*.

## Public Functions

```
std::shared_ptr<ConfigurationValue> registerConfigValue (QString name, QVariant deflt,
                                                         ConfigurationValueType *value-
                                                         type = 0, QString desc = "", Con-
                                                         figurationCategory cat = Catego-
                                                         ryNone, QString longdesc = "",
                                                         QString changehint = "")
```

Creates and registers a new *ConfigurationValue*. This is typically done once at the beginning. Each registered instance is shown in the ‘Tuning’ dialog and can be set (for making changes) and observed (for applying changes) in code as well. Useful e.g. for some internal bookkeeping or for exotic machine-wide configuration values (path to some tool, ...).

`QList<std::shared_ptr<ConfigurationValue>> getAllConfigurationValues ()`  
Returns a list of all registered configuration value instances.

`void doInitialize ()`  
Executes singleton initialization.

`void doShutdown ()`  
Executes singleton shutdown.

`void shutdown ()`  
Shutdown down this singleton.

`bool isAlive ()`  
Returns if this singleton is alive (true until its shutdown begins).

### Private Functions

`QVariant getFixedConfigValue (QString key)`  
Returns a non-null value if ‘key’ was set in a fixed way (e.g. via command line).

`ConfigurationManager ()`

### Private Members

`QHash<QString, std::shared_ptr<ConfigurationValue>> _configvalues`  
`QHash<QString, QVariant> _fixed_configvalues`  
`bool _fixed_configvalues_initialized = false`  
`QMutex _mutex`  
`class ConfigurationValueImpl : public sh::configuration::ConfigurationValue`

### Public Functions

`ConfigurationValueImpl (QString name, QVariant deft, QString desc, QString longdesc,  
ConfigurationCategory cat, ConfigurationValueType *value-  
type, QString changehint, ConfigurationManager *mgr)`

`void setConfigValue (QVariant value) override`  
Sets a new configuration value.

`QVariant getConfigValueVariant () override`  
Returns the current value of this instance as variant.

`bool mayChange () override`  
Returns if changes are allowed.

`int getConfigValueInt ()`  
Returns the current value of this instance as integer.

`double getConfigValueFloat ()`  
Returns the current value of this instance as floating point number.

`bool getConfigValueBool ()`  
Returns the current value of this instance as boolean.

`QString getConfigValueString ()`  
Returns the current value of this instance as string.

void **consumeValue** (QObject \*o, std::function<void>  
> h) Adds a consumer function, triggered directly and whenever the value changes.

#### Parameters

- o: For lifetime monitoring.

QString **name** ()

Returns the internal name of this configuration.

QString **description** ()

Returns the short description of this configuration.

QString **longDescription** ()

Returns the long description of this configuration.

*ConfigurationCategory* **category** ()

Returns the category of this configuration (mainly for UI structuring).

QString **changeHint** ()

Returns the change hint text of this configuration.

QVariant **defaultValue** ()

Returns the default value of this configuration.

*ConfigurationValueType* \***valueType** ()

Returns the value type of this configuration.

### Public Static Functions

*ConfigurationValueType* \***valueTypeString** ()

Returns the string type for configuration values.

*ConfigurationValueType* \***valueTypeInteger** ()

Returns the integer type for configuration values.

*ConfigurationValueType* \***valueTypeFloat** ()

Returns the floating point number type for configuration values.

*ConfigurationValueType* \***valueTypeBoolean** ()

Returns the boolean type for configuration values.

*ConfigurationValueType* \***valueTypeLocalFilePath** ()

Returns the filepath type for configuration values.

### Private Members

*ConfigurationManager* \*mgr

**class ConfigurationValue**

*#include <configurationvalue.h>* Abstract base class for a configuration value which is managed in the Tuning dialog.

Each instance can get and set the value associated to it.

You should typically not need to override it. See *sh::configuration::ConfigurationManager*. See Shallot documentation for more details.

Subclassed by *sh::configuration::ConfigurationManager::ConfigurationValueImpl*

## Public Functions

**ConfigurationValue** (QString *name*, QVariant *deflt*, QString *desc*, QString *longdesc*, *ConfigurationCategory* *cat*, *ConfigurationValueType* \**valuetype*, QString *changehint*)

int **getConfigValueInt** ()  
Returns the current value of this instance as integer.

double **getConfigValueFloat** ()  
Returns the current value of this instance as floating point number.

bool **getConfigValueBool** ()  
Returns the current value of this instance as boolean.

QString **getConfigValueString** ()  
Returns the current value of this instance as string.

QVariant **getConfigValueVariant** () = 0  
Returns the current value of this instance as variant.

void **setConfigValue** (QVariant *value*) = 0  
Sets a new configuration value.

bool **mayChange** () = 0  
Returns if changes are allowed.

void **consumeValue** (QObject \**o*, std::function<void>  
> *h*) Adds a consumer function, triggered directly and whenever the value changes.

### Parameters

- *o*: For lifetime monitoring.

QString **name** ()  
Returns the internal name of this configuration.

QString **description** ()  
Returns the short description of this configuration.

QString **longDescription** ()  
Returns the long description of this configuration.

*ConfigurationCategory* **category** ()  
Returns the category of this configuration (mainly for UI structuring).

QString **changeHint** ()  
Returns the change hint text of this configuration.

QVariant **defaultValue** ()  
Returns the default value of this configuration.

*ConfigurationValueType* \***valueType** ()  
Returns the value type of this configuration.

## Public Static Functions

*ConfigurationValueType* \***valueTypeString** ()

Returns the string type for configuration values.

*ConfigurationValueType* \***valueTypeInteger** ()

Returns the integer type for configuration values.

*ConfigurationValueType* \***valueTypeFloat** ()

Returns the floating point number type for configuration values.

*ConfigurationValueType* \***valueTypeBoolean** ()

Returns the boolean type for configuration values.

*ConfigurationValueType* \***valueTypeLocalFilePath** ()

Returns the filepath type for configuration values.

## Friends

**friend class** ConfigurationManager

**friend class** ConfigurationValueType

**class** ConfigurationValueType

*#include <configurationvaluetype.h>* Abstract base class for a configuration value type.

Each subclass implements support for a certain type of configuration values (e.g. string, integer, ...).

Subclassed by *sh::configuration::ConfigurationValueTypeBoolean*, *sh::configuration::ConfigurationValueTypeFloat*, *sh::configuration::ConfigurationValueTypeInteger*, *sh::configuration::ConfigurationValueTypeLocalFilePath*, *sh::configuration::ConfigurationValueTypeString*

## Public Functions

QString **valueDescription** (QVariant v)

QVariant **parse** (QString input)

**class** ConfigurationValueTypeBoolean : public *sh::configuration::ConfigurationValueType*

*#include <configurationvaluetype.h>* Configuration value type 'boolean'.

## Public Functions

QString **valueDescription** (QVariant v)

QVariant **parse** (QString input)

**class** ConfigurationValueTypeFloat : public *sh::configuration::ConfigurationValueType*

*#include <configurationvaluetype.h>* Configuration value type 'float'.

### Public Functions

QString **valueDescription** (QVariant v)

QVariant **parse** (QString input)

**class ConfigurationValueTypeInteger : public *sh::configuration::ConfigurationValueType***  
*#include <configurationvaluetype.h>* Configuration value type 'integer'.

### Public Functions

QString **valueDescription** (QVariant v)

QVariant **parse** (QString input)

**class ConfigurationValueTypeLocalFilePath : public *sh::configuration::ConfigurationValueType***  
*#include <configurationvaluetype.h>* Configuration value type 'file path'.

### Public Functions

QString **valueDescription** (QVariant v)

QVariant **parse** (QString input)

**class ConfigurationValueTypeString : public *sh::configuration::ConfigurationValueType***  
*#include <configurationvaluetype.h>* Configuration value type 'string'.

### Public Functions

QString **valueDescription** (QVariant v)

QVariant **parse** (QString input)

## 10.2.8 Namespace *sh::detailcolumns*

**namespace *sh::detailcolumns***

Implementations of detail columns.

Subclasses of *sh::filesystem::DetailColumn* (and possibly some auxiliary stuff). They are shown in the fileviews and can be queried in code.

**class DetailColumnCustomAttributes : public *sh::filesystem::DetailColumn*, public *sh::base::Singleton***  
*#include <detailcolumncustomattributes.h>* Detail column which determines the custom attributes.

Custom attributes are filesystem handler specific values about files (like permissions).

## Public Functions

void **applyValueByEurl** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*,  
*sh::filesystem::Operation* \**op*, QVariant *value*) **override**

QString **name** ()

The internal unique name.

QString **displayName** ()

The display name.

bool **isValueAvailable** (std::shared\_ptr<FilesystemNode> *node*)

Checks if a value for this detail is available for one given node.

QVariant **value** (std::shared\_ptr<FilesystemNode> *node*, bool *ignoreAged* = false)

Returns the value for this detail for one given node.

### Parameters

- *ignoreAged*: If no value should be returned which is older than the last request (not useful for nearly all cases).

QString **displayValue** (std::shared\_ptr<FilesystemNode> *node*, *const*  
*sh::filesystem::FilesystemModelFileviewProxy* \**viewmodel*)

Returns the stringified value for this details for one given node considering the configuration of a view.

QVariant **requestValue** (std::shared\_ptr<FilesystemNode> *node*, *sh::filesystem::Operation* \**op* =  
0, bool *withNodeValues* = false, bool *withOperationsCache* = true)

Requests to determine the value for this detail for one given node. Blocks until the value is available. Get it with the value method afterwards.

bool **sort\_doTypediff** ()

If sorting should separate files and directories.

int **sort\_order** (std::shared\_ptr<FilesystemNode> *n1*, std::shared\_ptr<FilesystemNode> *n2*)

The sorting order.

uint **displayIndex** ()

Controls which place this column should get in the user interface.

QVariant **computeValue** (std::shared\_ptr<FilesystemNode> *node*, *sh::filesystem::Operation* \**op*)

Returns a freshly determined value for this column and the given node. It neither asks nor populates any caches or storages.

int **defaultWidth** ()

bool **isRightAligned** ()

void **applyValueByEurl** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*,  
*sh::filesystem::Operation* \**op*, QVariant *value*)

Applies the detail value to a given eurl (without using any caches).

Used for transferring some details when a file transfer occurs.

Optionally implement the one of the applyValueBy... methods and register the instance with *sh::filesystem::FilesystemOperation::addTransferrableDetailColumn*. For performance reasons, you should decide to implement applyValueByEurl.

void **applyValueByNode** (std::shared\_ptr<FilesystemNode> *node*, *sh::filesystem::Operation* \**op*,  
QVariant *value*)

Applies the detail value to a given node (without using any caches).

Used for transferring some details when a file transfer occurs.

Optionally implement the one of the `applyValueBy...` methods and register the instance with `sh::filesystem::FilesystemOperation::addTransferrableDetailColumn`. For performance reasons, you should decide to implement `applyValueByEurl`.

```
void doShutdown ()  
    Executes singleton shutdown.  
  
void shutdown ()  
    Shutdown down this singleton.  
  
bool isAlive ()  
    Returns if this singleton is alive (true until its shutdown begins).
```

### Public Static Functions

```
QMap<QString, QString> getMapByValue (QVariant val)  
    Converts the QVariant(QString) from native detail column representation to a QString/QString map.  
  
QVariant VALUE_UNAVAILABLE ()  
    A special value expressing that the value is not yet available.  
  
void registerKnownDetailColumn (QString name, std::shared_ptr<DetailColumn> column)  
std::shared_ptr<DetailColumn> findKnownDetailColumn (QString name)
```

### Public Static Attributes

```
const uint DISPLAYINDEX_CORE = 10000  
const uint DISPLAYINDEX_INTERESTING = 20000  
const uint DISPLAYINDEX_EXOTIC = 30000
```

### Private Functions

```
DetailColumnCustomAttributes ()  
  
class DetailColumnDirectSymlinkTarget : public sh::filesystem::DetailColumn, public sh::base::Singleton  
    #include <detailcolumndirectsymlinktarget.h> Detail column which determines the direct symlink target  
    (by resolving a link once).
```

### Public Functions

```
QString name ()  
    The internal unique name.  
  
QString displayName ()  
    The display name.  
  
bool isValueAvailable (std::shared_ptr<FilesystemNode> node)  
    Checks if a value for this detail is available for one given node.  
  
QVariant value (std::shared_ptr<FilesystemNode> node, bool ignoreAged = false)  
    Returns the value for this detail for one given node.
```

#### Parameters

- `ignoreAged`: If no value should be returned which is older than the last request (not useful for nearly all cases).



**QString displayValue** (std::shared\_ptr<FilesystemNode> *node*, **const** *sh::filesystem::FilesystemModelFileviewProxy \*viewmodel*)  
Returns the stringified value for this details for one given node considering the configuration of a view.

**QVariant requestValue** (std::shared\_ptr<FilesystemNode> *node*, *sh::filesystem::Operation \*op* = 0, bool *withNodeValues* = false, bool *withOperationsCache* = true)  
Requests to determine the value for this detail for one given node. Blocks until the value is available. Get it with the value method afterwards.

**bool sort\_doTypediff** ()  
If sorting should separate files and directories.

**int sort\_order** (std::shared\_ptr<FilesystemNode> *n1*, std::shared\_ptr<FilesystemNode> *n2*)  
The sorting order.

**uint displayIndex** ()  
Controls which place this column should get in the user interface.

**QVariant computeValue** (std::shared\_ptr<FilesystemNode> *node*, *sh::filesystem::Operation \*op*)  
Returns a freshly determined value for this column and the given node. It neither asks nor populates any caches or storages.

**int defaultWidth** ()

**bool isRightAligned** ()

**void applyValueByEurl** (std::shared\_ptr<**const** *sh::filesystem::Eurl*> *eurl*, *sh::filesystem::Operation \*op*, *QVariant value*)  
Applies the detail value to a given eurl (without using any caches).  
Used for transferring some details when a file transfer occurs.  
Optionally implement the one of the applyValueBy... methods and register the instance with *sh::filesystem::FilesystemOperation::addTransferrableDetailColumn*. For performance reasons, you should decide to implement applyValueByEurl.

**void applyValueByNode** (std::shared\_ptr<FilesystemNode> *node*, *sh::filesystem::Operation \*op*, *QVariant value*)  
Applies the detail value to a given node (without using any caches).  
Used for transferring some details when a file transfer occurs.  
Optionally implement the one of the applyValueBy... methods and register the instance with *sh::filesystem::FilesystemOperation::addTransferrableDetailColumn*. For performance reasons, you should decide to implement applyValueByEurl.

**void doShutdown** ()  
Executes singleton shutdown.

**void shutdown** ()  
Shutdown down this singleton.

**bool isAlive** ()  
Returns if this singleton is alive (true until its shutdown begins).

## Public Static Functions

QVariant **VALUE\_UNAVAILABLE** ()

A special value expressing that the value is not yet available.

void **registerKnownDetailColumn** (QString *name*, std::shared\_ptr<DetailColumn> *column*)

std::shared\_ptr<DetailColumn> **findKnownDetailColumn** (QString *name*)

## Public Static Attributes

const uint **DISPLAYINDEX\_CORE** = 10000

const uint **DISPLAYINDEX\_INTERESTING** = 20000

const uint **DISPLAYINDEX\_EXOTIC** = 30000

## Private Functions

**DetailColumnDirectSymlinkTarget** ()

**class DetailColumnExtendedAttributes** : public *sh::filesystem::DetailColumn*, public *sh::base::Singleton*  
#include <detailcolumnextendedattributes.h> Detail column which determines the extended attributes.

Extended attributes are a feature of some filesystems.

## Public Functions

void **applyValueByEurl** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*,  
*sh::filesystem::Operation* \**op*, QVariant *value*) **override**

QString **name** ()

The internal unique name.

QString **displayName** ()

The display name.

bool **isValueAvailable** (std::shared\_ptr<FilesystemNode> *node*)

Checks if a value for this detail is available for one given node.

QVariant **value** (std::shared\_ptr<FilesystemNode> *node*, bool *ignoreAged* = false)

Returns the value for this detail for one given node.

### Parameters

- *ignoreAged*: If no value should be returned which is older than the last request (not useful for nearly all cases).

QString **displayValue** (std::shared\_ptr<FilesystemNode> *node*, const  
*sh::filesystem::FilesystemModelFileviewProxy* \**viewmodel*)

Returns the stringified value for this details for one given node considering the configuration of a view.

QVariant **requestValue** (std::shared\_ptr<FilesystemNode> *node*, *sh::filesystem::Operation* \**op* =  
0, bool *withNodeValues* = false, bool *withOperationsCache* = true)

Requests to determine the value for this detail for one given node. Blocks until the value is available.  
Get it with the value method afterwards.

bool **sort\_doTypediff** ()

If sorting should separate files and directories.

int **sort\_order** (std::shared\_ptr<FilesystemNode> *n1*, std::shared\_ptr<FilesystemNode> *n2*)

The sorting order.

uint **displayIndex** ()

Controls which place this column should get in the user interface.

QVariant **computeValue** (std::shared\_ptr<FilesystemNode> *node*, *sh::filesystem::Operation* \**op*)

Returns a freshly determined value for this column and the given node. It neither asks nor populates any caches or storages.

int **defaultWidth** ()

bool **isRightAligned** ()

void **applyValueByEurl** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*,  
*sh::filesystem::Operation* \**op*, QVariant *value*)

Applies the detail value to a given eurl (without using any caches).

Used for transferring some details when a file transfer occurs.

Optionally implement the one of the applyValueBy... methods and register the instance with *sh::filesystem::FilesystemOperation::addTransferrableDetailColumn*. For performance reasons, you should decide to implement applyValueByEurl.

void **applyValueByNode** (std::shared\_ptr<FilesystemNode> *node*, *sh::filesystem::Operation* \**op*,  
QVariant *value*)

Applies the detail value to a given node (without using any caches).

Used for transferring some details when a file transfer occurs.

Optionally implement the one of the applyValueBy... methods and register the instance with *sh::filesystem::FilesystemOperation::addTransferrableDetailColumn*. For performance reasons, you should decide to implement applyValueByEurl.

void **doShutdown** ()

Executes singleton shutdown.

void **shutdown** ()

Shutdown down this singleton.

bool **isAlive** ()

Returns if this singleton is alive (true until its shutdown begins).

## Public Static Functions

QMap<QString, QByteArray> **getMapByValue** (QVariant *val*)

Converts the QVariant(QString) from native detail column representation to a QString/QString map.

QVariant **VALUE\_UNAVAILABLE** ()

A special value expressing that the value is not yet available.

void **registerKnownDetailColumn** (QString *name*, std::shared\_ptr<DetailColumn> *column*)

std::shared\_ptr<DetailColumn> **findKnownDetailColumn** (QString *name*)

### Public Static Attributes

```
const uint DISPLAYINDEX_CORE = 10000
const uint DISPLAYINDEX_INTERESTING = 20000
const uint DISPLAYINDEX_EXOTIC = 30000
```

### Private Functions

```
DetailColumnExtendedAttributes()
```

```
class DetailColumnFilesize : public sh::filesystem::DetailColumn, public sh::base::Singleton
#include <detailcolumnfilesize.h> Detail column which determines the file size.
```

### Public Functions

```
QString name()
```

The internal unique name.

```
QString displayName()
```

The display name.

```
bool isValueAvailable (std::shared_ptr<FilesystemNode> node)
```

Checks if a value for this detail is available for one given node.

```
QVariant value (std::shared_ptr<FilesystemNode> node, bool ignoreAged = false)
```

Returns the value for this detail for one given node.

#### Parameters

- ignoreAged: If no value should be returned which is older than the last request (not useful for nearly all cases).

```
bool isVisible()
```

If this detail shall be a visible column in the view.

```
QVariant requestValue (std::shared_ptr<FilesystemNode> node, sh::filesystem::Operation *op =
0, bool withNodeValues = false, bool withOperationsCache = true)
```

Requests to determine the value for this detail for one given node. Blocks until the value is available. Get it with the value method afterwards.

```
bool sort_doTypediff()
```

If sorting should separate files and directories.

```
int sort_order (std::shared_ptr<FilesystemNode> n1, std::shared_ptr<FilesystemNode> n2)
```

The sorting order.

```
uint displayIndex()
```

Controls which place this column should get in the user interface.

```
QVariant computeValue (std::shared_ptr<FilesystemNode> node, sh::filesystem::Operation *op)
```

Returns a freshly determined value for this column and the given node. It neither asks nor populates any caches or storages.

```
int defaultWidth()
```

```
bool isRightAligned()
```

```
void applyValueByEurl (std::shared_ptr<const sh::filesystem::Eurl> eurl,
sh::filesystem::Operation *op, QVariant value)
```

Applies the detail value to a given eurl (without using any caches).

Used for transferring some details when a file transfer occurs.

Optionally implement the one of the `applyValueBy...` methods and register the instance with `sh::filesystem::FilesystemOperation::addTransferrableDetailColumn`. For performance reasons, you should decide to implement `applyValueByEurl`.

```
void applyValueByNode (std::shared_ptr<FilesystemNode> node, sh::filesystem::Operation *op,
                      QVariant value)
```

Applies the detail value to a given node (without using any caches).

Used for transferring some details when a file transfer occurs.

Optionally implement the one of the `applyValueBy...` methods and register the instance with `sh::filesystem::FilesystemOperation::addTransferrableDetailColumn`. For performance reasons, you should decide to implement `applyValueByEurl`.

```
void doShutdown ()
```

Executes singleton shutdown.

```
void shutdown ()
```

Shutdown down this singleton.

```
bool isAlive ()
```

Returns if this singleton is alive (true until its shutdown begins).

## Public Static Functions

```
QVariant VALUE_UNAVAILABLE ()
```

A special value expressing that the value is not yet available.

```
void registerKnownDetailColumn (QString name, std::shared_ptr<DetailColumn> column)
```

```
std::shared_ptr<DetailColumn> findKnownDetailColumn (QString name)
```

## Public Static Attributes

```
const uint DISPLAYINDEX_CORE = 10000
```

```
const uint DISPLAYINDEX_INTERESTING = 20000
```

```
const uint DISPLAYINDEX_EXOTIC = 30000
```

## Private Functions

```
DetailColumnFilesize ()
```

```
class DetailColumnMimetype : public sh::filesystem::DetailColumn, public sh::base::Singleton
    #include <detailcolumnmimetype.h> Detail column which determines the file's mimetype.
```

## Public Functions

QString **name** ()

The internal unique name.

QString **displayName** ()

The display name.

bool **isValueAvailable** (std::shared\_ptr<FilesystemNode> *node*)

Checks if a value for this detail is available for one given node.

QVariant **value** (std::shared\_ptr<FilesystemNode> *node*, bool *ignoreAged* = false)

Returns the value for this detail for one given node.

### Parameters

- *ignoreAged*: If no value should be returned which is older than the last request (not useful for nearly all cases).

bool **isVisible** ()

If this detail shall be a visible column in the view.

QVariant **requestValue** (std::shared\_ptr<FilesystemNode> *node*, *sh::filesystem::Operation* \**op* = 0, bool *withNodeValues* = false, bool *withOperationsCache* = true)

Requests to determine the value for this detail for one given node. Blocks until the value is available. Get it with the value method afterwards.

bool **sort\_doTypediff** ()

If sorting should separate files and directories.

int **sort\_order** (std::shared\_ptr<FilesystemNode> *n1*, std::shared\_ptr<FilesystemNode> *n2*)

The sorting order.

uint **displayIndex** ()

Controls which place this column should get in the user interface.

QVariant **computeValue** (std::shared\_ptr<FilesystemNode> *node*, *sh::filesystem::Operation* \**op*)

Returns a freshly determined value for this column and the given node. It neither asks nor populates any caches or storages.

int **defaultWidth** ()

bool **isRightAligned** ()

void **applyValueByEurl** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*,  
*sh::filesystem::Operation* \**op*, QVariant *value*)

Applies the detail value to a given eurl (without using any caches).

Used for transferring some details when a file transfer occurs.

Optionally implement the one of the applyValueBy... methods and register the instance with *sh::filesystem::FilesystemOperation::addTransferrableDetailColumn*. For performance reasons, you should decide to implement applyValueByEurl.

void **applyValueByNode** (std::shared\_ptr<FilesystemNode> *node*, *sh::filesystem::Operation* \**op*,  
QVariant *value*)

Applies the detail value to a given node (without using any caches).

Used for transferring some details when a file transfer occurs.

Optionally implement the one of the applyValueBy... methods and register the instance with *sh::filesystem::FilesystemOperation::addTransferrableDetailColumn*. For performance reasons, you should decide to implement applyValueByEurl.

void **doShutdown** ()  
 Executes singleton shutdown.

void **shutdown** ()  
 Shutdown down this singleton.

bool **isAlive** ()  
 Returns if this singleton is alive (true until its shutdown begins).

## Public Static Functions

QVariant **VALUE\_UNAVAILABLE** ()  
 A special value expressing that the value is not yet available.

void **registerKnownDetailColumn** (QString *name*, std::shared\_ptr<DetailColumn> *column*)  
 std::shared\_ptr<DetailColumn> **findKnownDetailColumn** (QString *name*)

## Public Static Attributes

const uint **DISPLAYINDEX\_CORE** = 10000  
 const uint **DISPLAYINDEX\_INTERESTING** = 20000  
 const uint **DISPLAYINDEX\_EXOTIC** = 30000

## Private Functions

**DetailColumnMimetype** ()

**class DetailColumnMtime** : public *sh::filesystem::DetailColumn*, public *sh::base::Singleton*  
*#include <detailcolumnmtime.h>* Detail column which determines the file's modification time.

## Public Functions

void **applyValueByEurl** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*,  
*sh::filesystem::Operation* \**op*, QVariant *value*) **override**

QString **name** ()  
 The internal unique name.

QString **displayName** ()  
 The display name.

bool **isValueAvailable** (std::shared\_ptr<FilesystemNode> *node*)  
 Checks if a value for this detail is available for one given node.

QVariant **value** (std::shared\_ptr<FilesystemNode> *node*, bool *ignoreAged* = false)  
 Returns the value for this detail for one given node.

### Parameters

- *ignoreAged*: If no value should be returned which is older than the last request (not useful for nearly all cases).

bool **isVisible** ()  
 If this detail shall be a visible column in the view.

**QVariant requestValue** (std::shared\_ptr<FilesystemNode> *node*, *sh::filesystem::Operation* \**op* = 0, bool *withNodeValues* = false, bool *withOperationsCache* = true)  
Requests to determine the value for this detail for one given node. Blocks until the value is available. Get it with the value method afterwards.

bool **sort\_doTypediff** ()  
If sorting should separate files and directories.

int **sort\_order** (std::shared\_ptr<FilesystemNode> *n1*, std::shared\_ptr<FilesystemNode> *n2*)  
The sorting order.

uint **displayIndex** ()  
Controls which place this column should get in the user interface.

**QVariant computeValue** (std::shared\_ptr<FilesystemNode> *node*, *sh::filesystem::Operation* \**op*)  
Returns a freshly determined value for this column and the given node. It neither asks nor populates any caches or storages.

int **defaultWidth** ()

bool **isRightAligned** ()

void **applyValueByEurl** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*, *sh::filesystem::Operation* \**op*, QVariant *value*)  
Applies the detail value to a given eurl (without using any caches).  
  
Used for transferring some details when a file transfer occurs.  
  
Optionally implement the one of the applyValueBy... methods and register the instance with *sh::filesystem::FilesystemOperation::addTransferrableDetailColumn*. For performance reasons, you should decide to implement applyValueByEurl.

void **applyValueByNode** (std::shared\_ptr<FilesystemNode> *node*, *sh::filesystem::Operation* \**op*, QVariant *value*)  
Applies the detail value to a given node (without using any caches).  
  
Used for transferring some details when a file transfer occurs.  
  
Optionally implement the one of the applyValueBy... methods and register the instance with *sh::filesystem::FilesystemOperation::addTransferrableDetailColumn*. For performance reasons, you should decide to implement applyValueByEurl.

void **doShutdown** ()  
Executes singleton shutdown.

void **shutdown** ()  
Shutdown down this singleton.

bool **isAlive** ()  
Returns if this singleton is alive (true until its shutdown begins).

## Public Static Functions

**QVariant VALUE\_UNAVAILABLE** ()  
A special value expressing that the value is not yet available.

void **registerKnownDetailColumn** (QString *name*, std::shared\_ptr<DetailColumn> *column*)

std::shared\_ptr<DetailColumn> **findKnownDetailColumn** (QString *name*)



## Public Static Attributes

```
const uint DISPLAYINDEX_CORE = 10000
const uint DISPLAYINDEX_INTERESTING = 20000
const uint DISPLAYINDEX_EXOTIC = 30000
```

## Private Functions

```
DetailColumnMtime()
```

**class** **DetailColumnResolvedSymlink** : **public** *sh::filesystem::DetailColumn*, **public** *sh::base::Singleton*  
*#include <detailcolumnresolvedsymlink.h>* Detail column which determines the symlink target (by resolving links recursively).

This detail column has some active behavior. It informs the filesystem model about the ‘buddy nodes’.

## Public Functions

```
QString name()
    The internal unique name.
```

```
QString displayName()
    The display name.
```

```
bool isValueAvailable(std::shared_ptr<FilesystemNode> node)
    Checks if a value for this detail is available for one given node.
```

```
QVariant value(std::shared_ptr<FilesystemNode> node, bool ignoreAged = false)
    Returns the value for this detail for one given node.
```

### Parameters

- *ignoreAged*: If no value should be returned which is older than the last request (not useful for nearly all cases).

```
QVariant requestValue(std::shared_ptr<FilesystemNode> node, sh::filesystem::Operation *op =
    0, bool withNodeValues = false, bool withOperationsCache = true)
    Requests to determine the value for this detail for one given node. Blocks until the value is available.
    Get it with the value method afterwards.
```

```
bool sort_doTypediff()
    If sorting should separate files and directories.
```

```
int sort_order(std::shared_ptr<FilesystemNode> n1, std::shared_ptr<FilesystemNode> n2)
    The sorting order.
```

```
uint displayIndex()
    Controls which place this column should get in the user interface.
```

```
QVariant computeValue(std::shared_ptr<FilesystemNode> node, sh::filesystem::Operation *op)
    Returns a freshly determined value for this column and the given node. It neither asks nor populates
    any caches or storages.
```

```
int defaultWidth()
```

```
bool isRightAligned()
```

```
void applyValueByEurl(std::shared_ptr<const sh::filesystem::Eurl> eurl,
    sh::filesystem::Operation *op, QVariant value)
    Applies the detail value to a given eurl (without using any caches).
```

Used for transferring some details when a file transfer occurs.

Optionally implement the one of the `applyValueBy...` methods and register the instance with `sh::filesystem::FilesystemOperation::addTransferrableDetailColumn`. For performance reasons, you should decide to implement `applyValueByEurl`.

```
void applyValueByNode (std::shared_ptr<FilesystemNode> node, sh::filesystem::Operation *op,  
                      QVariant value)
```

Applies the detail value to a given node (without using any caches).

Used for transferring some details when a file transfer occurs.

Optionally implement the one of the `applyValueBy...` methods and register the instance with `sh::filesystem::FilesystemOperation::addTransferrableDetailColumn`. For performance reasons, you should decide to implement `applyValueByEurl`.

```
void doShutdown ()
```

Executes singleton shutdown.

```
void shutdown ()
```

Shutdown down this singleton.

```
bool isAlive ()
```

Returns if this singleton is alive (true until its shutdown begins).

### Public Static Functions

```
QVariant VALUE_UNAVAILABLE ()
```

A special value expressing that the value is not yet available.

```
void registerKnownDetailColumn (QString name, std::shared_ptr<DetailColumn> column)
```

```
std::shared_ptr<DetailColumn> findKnownDetailColumn (QString name)
```

### Public Static Attributes

```
const uint DISPLAYINDEX_CORE = 10000
```

```
const uint DISPLAYINDEX_INTERESTING = 20000
```

```
const uint DISPLAYINDEX_EXOTIC = 30000
```

### Private Functions

```
DetailColumnResolvedSymlink ()
```

## 10.2.9 Namespace `sh::exceptions`

```
namespace sh::exceptions
```

Exceptions.

The class hierarchy of Shallot exceptions as well as some utilities for exception handling.

## Enums

**enum ExecuteGuardFlag**

*Values:*

**enumerator** `MANUAL_RETRY_ENABLED` = 1 << 0

**enumerator** `AUTOMATIC_RETRY_ENABLED` = 1 << 1

**enumerator** `LOGGING_DISABLED` = 1 << 2

**enumerator** `RESUMEABLE_PROGRAM_ERROR_END_WITH_USER_FEEDBACK_HERE` = 1 << 3

**enumerator** `CANCELABLE_UP_TO_HERE` = 1 << 4

**enumerator** `ALL_ERRORS_KILL_SHALLOT` = 1 << 5

**enumerator** `IGNORE_ALL_RESUMEABLE_ERRORS_SILENTLY` = 1 << 6

**class** `ArgumentException`: public *sh::exceptions::ProgramException*

*#include <argumentexception.h>* Shallot exception for failed operation due to invalid arguments given to some program part. It typically allows resume but not retry (special cases may override each of them).

Subclassed by *sh::filesystem::EurlMisformattedException*

## Public Functions

**ArgumentException** (*ExceptionData* data)

**QString** `message` () const

**QString** `name` () const

**QString** `classes` () const

**QString** `details` () const

**QString** `callstack` () const

**QString** `auxiliary` () const

**QString** `customValue` (QString key) const

**bool** `isRuntimeException` () const

**bool** `isProgramException` () const

**bool** `isRetryable` () const

**bool** `isResumeable` () const

**bool** `isDetailsAreInteresting` () const

**int** `autoRetryRecommendedIn` () const

**void** `setResumeable` (bool v)

**void** `setReTryable` (bool v)

**void** `setCustomValue` (QString key, QString value)

**bool** `isClass` (QString classname)

*sh::exceptions::ExceptionData* `data` ()

## Public Static Functions

```
template<class Handler>
std::shared_ptr<RegisterHandler<Handler>> createRegisterHandler (Handler handler,
                                                                QStack<Handler>
                                                                *stack)
```

```
void executeGuarded (std::function<void>
    > fct int flags = 0) Executes some code with some standard exection handling around.
```

### Parameters

- *fct*: The code to execute guarded.
- *flags*: Flags of *ExecuteGuardFlag* for choosing the behavior.

```
void executeGuarded_errorpanel (std::function<void>
    > fct int flags = 0)
```

```
QString _demangle (QString l)
```

```
void exceptionDialog (sh::exceptions::Exception &e, bool *doRetry)
```

## Public Static Attributes

```
const QString UNDEFINED_ERROR_OCCURRED = QObject::tr("An unspecified error occurred.")
```

```
const QString SHALLOT_MUST_CLOSE_TEXT = QObject::tr("Your Shallot process is disturbed by an error and needs to be closed.")
```

```
const QString Value_isShallotException = "_isShallotException"
```

## class CancelException

*#include <exception.h>* A special exception for cancellation of some action on user behalf.

A very special exception outside of the hierarchy. It is only used by the infrastructure; never throw it directly.

## Public Functions

```
CancelException ()
```

## class Exception

*#include <exception.h>* Shallot exception base class. Also contains some static methods for general work with exceptions.

Subclassed by *sh::exceptions::ProgramException*, *sh::exceptions::RuntimeException*, *sh::scripting::ScriptingEngine::ScriptedException*

## Public Functions

```
QString message () const
```

```
QString name () const
```

```
QString classes () const
```

```
QString details () const
```

```
QString callstack () const
```

```
QString auxiliary () const
```

```
QString customValue (QString key) const
```

```

bool isRuntimeException() const
bool isProgramException() const
bool isRetryable() const
bool isResumeable() const
bool isDetailsAreInteresting() const
int autoRetryRecommendedIn() const
void setResumeable(bool v)
void setReTryable(bool v)
void setCustomValue(QString key, QString value)
bool isClass(QString classname)
Exception(ExceptionData data)
Exception()
sh::exceptions::ExceptionData data()

```

## Public Static Functions

```

template<class Handler>
std::shared_ptr<RegisterHandler<Handler>> createRegisterHandler(Handler handler,
                                                                QStack<Handler>
                                                                *stack)

void executeGuarded(std::function<void>
    > fct int flags = 0) Executes some code with some standard exection handling around.

    Parameters
    • fct: The code to execute guarded.
    • flags: Flags of ExecuteGuardFlag for choosing the behavior.

void executeGuarded_errorpanel(std::function<void>
    > fct int flags = 0

QString _demangle(QString l)

void exceptionDialog(sh::exceptions::Exception &e, bool *doRetry)

```

## Public Static Attributes

```

const QString UNDEFINED_ERROR_OCCURRED = QObject::tr("An unspecified error occurred.")
const QString SHALLOT_MUST_CLOSE_TEXT = QObject::tr("Your Shallot process is disturbed by an error and needs to be restarted.")
const QString Value_isShallotException = "_isShallotException"

```

### Private Static Functions

```
void exceptionDialog (QString error1, QString error2, QString details, QString icon, bool  
    mayRetry, bool mayClose, bool mayCancel, bool showLoglabelAndDe-  
    tails, bool *doRetry)
```

### Private Static Attributes

```
bool _inShutdown = false  
QMutex _inShutdownMutex  
class HandlerSettings  
    #include <exception.h>
```

### Public Members

```
QString cancelText  
QStack<std::function<bool (sh::exceptions::Exception&) >> exceptionHandler_retryable  
QStack<std::function<void (sh::exceptions::Exception&) >> exceptionHandler_resumeable  
QStack<std::function<void (sh::exceptions::Exception&) >> exceptionHandler_hard  
QStack<std::function<void () >> exceptionHandler_cancel
```

### Private Functions

```
HandlerSettings ()
```

### Friends

```
friend class ExceptionHandlerSettingsManager  
template<class Handler>  
class RegisterHandler  
    #include <exception.h>
```

### Public Functions

```
RegisterHandler (Handler handler, QStack<Handler> *stack)  
~RegisterHandler ()
```

## Private Members

QStack<*Handler*> \*\_stack

**class ExceptionData** : public QMap<QString, QString>  
*#include <exception.h>* Used for specifying metadata for a *sh::exceptions::Exception*.

## Public Functions

**ExceptionData** (const *ExceptionData* &o)

**ExceptionData** (const QMap<QString, QString> &o)

**ExceptionData** ()

*ExceptionData* **name** (QString name)

*ExceptionData* **classes** (QString classes)

*ExceptionData* **aux** (QString aux)

*ExceptionData* **details** (QString details)

*ExceptionData* **message** (QString message)

*ExceptionData* **runtime** ()

*ExceptionData* **program** ()

*ExceptionData* **retryable** (bool v = true)

*ExceptionData* **resumeable** (bool v = true)

*ExceptionData* **autoRetryRecommended** (int v = -1)

*ExceptionData* **detailsAreInteresting** (bool v = true)

**class ExceptionHandlerSettingsManager** : public *sh::base::Singleton*  
*#include <exception.h>* Managing how to default-handle unhandled exceptions.

## Public Functions

*Exception::HandlerSettings* \***handlerSettings** ()  
 Returns the current *Exception::HandlerSettings*.

**~ExceptionHandlerSettingsManager** ()

void **doInitialize** ()  
 Executes singleton initialization.

void **doShutdown** ()  
 Executes singleton shutdown.

void **shutdown** ()  
 Shutdown down this singleton.

bool **isAlive** ()  
 Returns if this singleton is alive (true until its shutdown begins).

## Private Functions

**ExceptionHandlerSettingsManager** ()

## Private Members

QMap<QThread\*, *Exception::HandlerSettings\**> **\_handlerSettings**

QMux **\_mutex**

**class IOException**: public *sh::exceptions::RuntimeException*

*#include <ioexception.h>* Shallot exception in IO. It allows resume and typically allows retry (special cases may override each of them).

Subclassed by *sh::exceptions::PermissionDeniedException*, *sh::filesystem::Operation::MaxAllowedSizeRatioPerPartExceededException*

## Public Functions

**IOException** (*ExceptionData data*)

QString **message** () const

QString **name** () const

QString **classes** () const

QString **details** () const

QString **callstack** () const

QString **auxiliary** () const

QString **customValue** (QString *key*) const

bool **isRuntimeException** () const

bool **isProgramException** () const

bool **isRetryable** () const

bool **isResumeable** () const

bool **isDetailsAreInteresting** () const

int **autoRetryRecommendedIn** () const

void **setResumeable** (bool *v*)

void **setRetryable** (bool *v*)

void **setCustomValue** (QString *key*, QString *value*)

bool **isClass** (QString *classname*)

*sh::exceptions::ExceptionData data* ()



## Public Static Functions

```
template<class Handler>
std::shared_ptr<RegisterHandler<Handler>> createRegisterHandler (Handler handler,
                                                                QStack<Handler>
                                                                *stack)
```

```
void executeGuarded (std::function<void>
    > fct; int flags = 0) Executes some code with some standard execution handling around.
```

### Parameters

- *fct*: The code to execute guarded.
- *flags*: Flags of *ExecuteGuardFlag* for choosing the behavior.

```
void executeGuarded_errorpanel (std::function<void>
    > fct; int flags = 0)
```

```
QString _demangle (QString l)
```

```
void exceptionDialog (sh::exceptions::Exception &e, bool *doRetry)
```

## Public Static Attributes

```
const QString UNDEFINED_ERROR_OCCURRED = QObject::tr("An unspecified error occurred.")
```

```
const QString SHALLOT_MUST_CLOSE_TEXT = QObject::tr("Your Shallot process is disturbed by an error and needs to be closed.")
```

```
const QString Value_isShallotException = "_isShallotException"
```

```
class PermissionDeniedException : public sh::exceptions::IOException
    #include <permissiondeniedexception.h> Shallot exception for something forbidden was tried to execute.
    It allows resume and retry (special cases may override each of them).
```

## Public Functions

```
PermissionDeniedException (ExceptionData data)
```

```
QString message () const
```

```
QString name () const
```

```
QString classes () const
```

```
QString details () const
```

```
QString callstack () const
```

```
QString auxiliary () const
```

```
QString customValue (QString key) const
```

```
bool isRuntimeException () const
```

```
bool isProgramException () const
```

```
bool isRetryable () const
```

```
bool isResumeable () const
```

```
bool isDetailsAreInteresting () const
```

```
int autoRetryRecommendedIn () const
```

```
void setResumeable (bool v)
```

```
void setReTryable (bool v)
void setCustomValue (QString key, QString value)
bool isClass (QString classname)
sh::exceptions::ExceptionData data ()
```

### Public Static Functions

```
template<class Handler>
std::shared_ptr<RegisterHandler<Handler>> createRegisterHandler (Handler handler,
                                                             QStack<Handler>
                                                             *stack)
```

```
void executeGuarded (std::function<void>
    > fc int flags = 0) Executes some code with some standard exection handling around.
```

#### Parameters

- *fc*: The code to execute guarded.
- *flags*: Flags of *ExecuteGuardFlag* for choosing the behavior.

```
void executeGuarded_errorpanel (std::function<void>
    > fc int flags = 0)
```

```
QString _demangle (QString l)
```

```
void exceptionDialog (sh::exceptions::Exception &e, bool *doRetry)
```

### Public Static Attributes

```
const QString UNDEFINED_ERROR_OCCURRED = QObject::tr("An unspecified error occurred.")
```

```
const QString SHALLOT_MUST_CLOSE_TEXT = QObject::tr("Your Shallot process is disturbed by an error and needs to be closed.")
```

```
const QString Value_isShallotException = "_isShallotException"
```

```
class ProgramException: public sh::exceptions::Exception
    #include <programexception.h> Shallot exception for internal bugs in Shallot. It optionally allows resume
    but no retry (special cases may override each of them).
```

Subclassed by *sh::exceptions::ArgumentException*, *sh::exceptions::ThreadingException*

### Public Functions

```
ProgramException (ExceptionData data)
QString message () const
QString name () const
QString classes () const
QString details () const
QString callstack () const
QString auxiliary () const
QString customValue (QString key) const
bool isRuntimeException () const
```

```

bool isProgramException() const
bool isRetryable() const
bool isResumeable() const
bool isDetailsAreInteresting() const
int autoRetryRecommendedIn() const
void setResumeable(bool v)
void setReTryable(bool v)
void setCustomValue(QString key, QString value)
bool isClass(QString classname)
sh::exceptions::ExceptionData data()

```

### Public Static Functions

```

template<class Handler>
std::shared_ptr<RegisterHandler<Handler>> createRegisterHandler(Handler handler,
                                                             QStack<Handler>
                                                             *stack)

void executeGuarded(std::function<void>
    > fct int flags = 0) Executes some code with some standard exection handling around.

Parameters
    • fct: The code to execute guarded.
    • flags: Flags of ExecuteGuardFlag for choosing the behavior.

void executeGuarded_errorpanel(std::function<void>
    > fct int flags = 0

QString _demangle(QString l)

void exceptionDialog(sh::exceptions::Exception &e, bool *doRetry)

```

### Public Static Attributes

```

const QString UNDEFINED_ERROR_OCCURRED = QObject::tr("An unspecified error occurred.")
const QString SHALLOT_MUST_CLOSE_TEXT = QObject::tr("Your Shallot process is disturbed by an error and needs to be closed.")
const QString Value_isShallotException = "_isShallotException"

class RuntimeException: public sh::exceptions::Exception
    #include <runtimeexception.h> Shallot exception for failed operation due to (often external) runtime effects. It allows resume and optionally allows retry (special cases may override each of them).

    Subclassed by sh::exceptions::IOException

```

## Public Functions

```
RuntimeException (ExceptionData data)
QString message () const
QString name () const
QString classes () const
QString details () const
QString callstack () const
QString auxiliary () const
QString customValue (QString key) const
bool isRuntimeException () const
bool isProgramException () const
bool isRetryable () const
bool isResumeable () const
bool isDetailsAreInteresting () const
int autoRetryRecommendedIn () const
void setResumeable (bool v)
void setReTryable (bool v)
void setCustomValue (QString key, QString value)
bool isClass (QString classname)
sh::exceptions::ExceptionData data ()
```

## Public Static Functions

```
template<class Handler>
std::shared_ptr<RegisterHandler<Handler>> createRegisterHandler (Handler handler,
                                                                QStack<Handler>
                                                                *stack)

void executeGuarded (std::function<void>
    > fct int flags = 0 Executes some code with some standard exection handling around.

Parameters
    • fct: The code to execute guarded.
    • flags: Flags of ExecuteGuardFlag for choosing the behavior.

void executeGuarded_errorpanel (std::function<void>
    > fct int flags = 0

QString _demangle (QString l)

void exceptionDialog (sh::exceptions::Exception &e, bool *doRetry)
```

## Public Static Attributes

**const** QString **UNDEFINED\_ERROR\_OCCURRED** = QObject::tr("An unspecified error occurred.")

**const** QString **SHALLOT\_MUST\_CLOSE\_TEXT** = QObject::tr("Your Shallot process is disturbed by an error and needs to be closed.")

**const** QString **Value\_isShallotException** = "\_isShallotException"

**class** ThreadAbortException

*#include <exception.h>* Only used by the infrastructure; never throw it directly.

## Public Functions

**ThreadAbortException**()

**class** ThreadingException : public *sh::exceptions::ProgramException*

*#include <threadingexception.h>* Shallot exception for misuse of threading (e.g. wrong caller thread). It does not allow resume (special cases may override each of them).

## Public Functions

**ThreadingException**(*ExceptionData* data)

QString **message**() **const**

QString **name**() **const**

QString **classes**() **const**

QString **details**() **const**

QString **callstack**() **const**

QString **auxiliary**() **const**

QString **customValue**(QString key) **const**

bool **isRuntimeException**() **const**

bool **isProgramException**() **const**

bool **isRetryable**() **const**

bool **isResumeable**() **const**

bool **isDetailsAreInteresting**() **const**

int **autoRetryRecommendedIn**() **const**

void **setResumeable**(bool v)

void **setRetryable**(bool v)

void **setCustomValue**(QString key, QString value)

bool **isClass**(QString classname)

*sh::exceptions::ExceptionData* **data**()

## Public Static Functions

```
template<class Handler>
std::shared_ptr<RegisterHandler<Handler>> createRegisterHandler (Handler handler,
                                                                QStack<Handler>
                                                                *stack)
```

```
void executeGuarded (std::function<void>
    > fct int flags = 0) Executes some code with some standard exection handling around.
```

### Parameters

- *fct*: The code to execute guarded.
- *flags*: Flags of *ExecuteGuardFlag* for choosing the behavior.

```
void executeGuarded_errorpanel (std::function<void>
    > fct int flags = 0)
```

```
QString _demangle (QString l)
```

```
void exceptionDialog (sh::exceptions::Exception &e, bool *doRetry)
```

## Public Static Attributes

```
const QString UNDEFINED_ERROR_OCCURRED = QObject::tr("An unspecified error occurred.")
```

```
const QString SHALLOT_MUST_CLOSE_TEXT = QObject::tr("Your Shallot process is disturbed by an error and needs to be restarted.")
```

```
const QString Value_isShallotException = "_isShallotException"
```

## 10.2.10 Namespace *sh::filepropertydialogtabs*

**namespace *sh::filepropertydialogtabs***

Implementations of tabs in a file property dialog.

Subclasses of *sh::ui::FilePropertyDialogTab* (and possibly some auxiliary stuff). They implement content in file property dialogs.

```
class FilePropertyDialogTabExtendedAttributes : public sh::ui::FilePropertyDialogTab
    #include <filepropertydialogtabextendedattributes.h> Properties dialog tab for extended attributes.
```

## Public Functions

```
FilePropertyDialogTabExtendedAttributes ()
```

```
QString title () override
```

The tab title. .

```
QList<QString> properties () override
```

Returns the list of property labels (one entry for each widget).

Each property is displayed with a header and a widget (created in *createWidget*) with some content (populated in *updateWidget*).

```
void populateWidget (int i, sh::ui::FilePropertyDialogTabActionsView *widget) override
```

Populates the tab widget for the *i*-th property.

Typically uses the *FilePropertyDialog::create\** methods to create sub-widgets.

See also *dialog()*.

void **updateWidget** (int *i*, *sh::ui::FilePropertyDialogTabActionsView* \**widget*,  
*sh::filesystem::Operation* \**op*) **override**  
 Populates the widget for the i-th property with actual content. .

QString **titleWithoutMnemonic** ()  
 The tab title without mnemonic.

void **refresh** ()  
 Refreshes the complete content.

Typically called after some user actions in a property widget require that all the other content must be refreshed.

QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> **nodes** ()  
 The nodes to show.

*FilePropertyDialogTabActionsView* \***widgetAt** (int *i*)  
 Returns the widget for the i-th property.

int **widgetCount** ()  
 Returns the number of widgets.

std::shared\_ptr<*FilePropertyDialog*> **dialog** ()  
 Returns the associated dialog.

## Public Static Functions

void **doInitialize** ()

void **doShutdown** ()

**class FilePropertyDialogTabGeneral** : public *sh::ui::FilePropertyDialogTab*  
*#include <filepropertydialogtabgeneral.h>* Properties dialog tab for general infos.

## Public Functions

**FilePropertyDialogTabGeneral** ()

QString **title** () **override**  
 The tab title. .

QList<QString> **properties** () **override**  
 Returns the list of property labels (one entry for each widget).

Each property is displayed with a header and a widget (created in createWidget) with some content (populated in updateWidget).

void **populateWidget** (int *i*, *sh::ui::FilePropertyDialogTabActionsView* \**widget*) **override**  
 Populates the tab widget for the i-th property.

Typically uses the *FilePropertyDialog::create\** methods to create sub-widgets.

See also *dialog()*.

void **updateWidget** (int *i*, *sh::ui::FilePropertyDialogTabActionsView* \**widget*,  
*sh::filesystem::Operation* \**op*) **override**  
 Populates the widget for the i-th property with actual content. .

QString **titleWithoutMnemonic** ()  
 The tab title without mnemonic.

void **refresh** ()

Refreshes the complete content.

Typically called after some user actions in a property widget require that all the other content must be refreshed.

QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> **nodes** ()

The nodes to show.

FilePropertyDialogTabActionsView **\*widgetAt** (int *i*)

Returns the widget for the i-th property.

int **widgetCount** ()

Returns the number of widgets.

std::shared\_ptr<FilePropertyDialog> **dialog** ()

Returns the associated dialog.

## Public Static Functions

void **doInitialize** ()

void **doShutdown** ()

**class FilePropertyDialogTabUnixPermissions** : public *sh::ui::FilePropertyDialogTab*  
*#include <filepropertydialogtabunixpermissions.h>* Properties dialog tab for unix permissions.

## Public Functions

**FilePropertyDialogTabUnixPermissions** ()

QString **title** () **override**

The tab title. .

QList<QString> **properties** () **override**

Returns the list of property labels (one entry for each widget).

Each property is displayed with a header and a widget (created in createWidget) with some content (populated in updateWidget).

void **populateWidget** (int *i*, *sh::ui::FilePropertyDialogTabActionsView* *\*widget*) **override**

Populates the tab widget for the i-th property.

Typically uses the FilePropertyDialog::create\* methods to create sub-widgets.

See also *dialog()*.

void **updateWidget** (int *i*, *sh::ui::FilePropertyDialogTabActionsView* *\*widget*,  
*sh::filesystem::Operation* *\*op*) **override**

Populates the widget for the i-th property with actual content. .

QString **titleWithoutMnemonic** ()

The tab title without mnemonic.

void **refresh** ()

Refreshes the complete content.

Typically called after some user actions in a property widget require that all the other content must be refreshed.

QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> **nodes** ()

The nodes to show.



FilePropertyDialogTabActionsView \*widgetAt (int i)  
Returns the widget for the i-th property.

int widgetCount ()  
Returns the number of widgets.

std::shared\_ptr<FilePropertyDialog> dialog ()  
Returns the associated dialog.

## Public Static Functions

QString getUsername (int uid)

QString getGroupName (int gid)

QMap<int, QString> getAllUsers ()

QMap<int, QString> getAllGroups ()

void doInitialize ()

void doShutdown ()

## Private Functions

void slot\_buttontriggered (int i)

QString userPermissionsDescription (int flags, int rflag, int wflag, int xflag)

class ActionChange : public sh::actions::ActionActionItem

## Public Functions

ActionChange (std::shared\_ptr<sh::filepropertydialogtabs::FilePropertyDialogTabUnixPermissions>  
tab)

void execute ()  
Executes this action.

void execute (sh::actions::ActionExecutionInfo \*info)  
Executes this action.

QKeySequence shortcutHint ()  
Returns the keyboard shortcut for triggering this action.

bool shortcutHintTriggersOnCurrentDirectory ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void setShortcutHint (QKeySequence shortcut, bool triggersOnCurrentDirectory = false)  
Sets the keyboard shortcut for triggering this action.

QString text ()  
Returns the displayed text for this action.

QString icon ()  
Returns the icon for this action.

bool enabled ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<AbstractActionItem> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_\_setParentAction** (std::shared\_ptr<AbstractActionItem> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

**class FilePropertyDialogTabWindows : public *sh::ui::FilePropertyDialogTab***  
*#include <filepropertydialogtabwindows.h>* Properties dialog tab for unix permissions.

## Public Functions

QString **title** () = 0

The tab title. .

QString **titleWithoutMnemonic** ()

The tab title without mnemonic.

QList<QString> **properties** () = 0

Returns the list of property labels (one entry for each widget).

Each property is displayed with a header and a widget (created in `createWidget`) with some content (populated in `updateWidget`).

void **populateWidget** (int *i*, FilePropertyDialogTabActionsView \**widget*) = 0

Populates the tab widget for the *i*-th property.

Typically uses the `FilePropertyDialog::create*` methods to create sub-widgets.

See also `dialog()`.

void **updateWidget** (int *i*, FilePropertyDialogTabActionsView \**widget*, *sh::filesystem::Operation* \**op*) = 0

Populates the widget for the *i*-th property with actual content. .

void **refresh** ()

Refreshes the complete content.

Typically called after some user actions in a property widget require that all the other content must be refreshed.

QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> **nodes** ()

The nodes to show.

FilePropertyDialogTabActionsView \***widgetAt** (int *i*)

Returns the widget for the *i*-th property.

int **widgetCount** ()

Returns the number of widgets.

std::shared\_ptr<FilePropertyDialog> **dialog** ()

Returns the associated dialog.

## 10.2.11 Namespace `sh::filesystem`

namespace `sh::filesystem`

Filesystem model and auxiliary logic.

### Enums

enum **SizeFormatting**

Values:

enumerator **SizeFormattingModePrefixes**

enumerator **SizeFormattingModePlainBytes**

enum **FilesystemNodeType**

Enumeration of types a `sh::filesystem::FilesystemNode` can have.

Values:

**enumerator NONE** = 0

No type specified or does not exist.

**enumerator File** = 1

Usual file.

**enumerator FIRSTTYPE** = *File*

The first type (used for generating int ranges).

**enumerator Directory**

Directory.

**enumerator Link**

Link.

**enumerator Unknown**

Unknown.

**enumerator LASTPHYSICALTYPE** = *Unknown*

The last physical type, excluding internal magic types (used for generating int ranges).

**enumerator SpecialTreeOnlyDirectory**

Special kind of directory which behaves differently.

**enumerator Invalid**

This return type means that the type can't be determined. It should not occur in typical situations.

**enumerator LASTTYPE** = *SpecialTreeOnlyDirectory*

The last type (used for generating int ranges).

**class AdhocFilesystemNodeList** : public *sh::filesystem::FilesystemNodeList*

*#include <filesystemodelist.h>* This *FilesystemNodeList* subclass is used for spontaneously fetching a fresh list of node children.

Node additions and removals will not touch the model.

Node instances inside it will be deleted together with this list.

## Public Functions

**AdhocFilesystemNodeList** ()

Is intended to be directly constructed from everywhere.

**~AdhocFilesystemNodeList** ()

void **addItem** (QSet<std::shared\_ptr<*FilesystemNode*>> *nodes*)

void **addItem** (std::shared\_ptr<*FilesystemNode*> *node*)

void **removeItems** (QSet<std::shared\_ptr<*FilesystemNode*>> *nodes*)

void **removeItem** (std::shared\_ptr<*FilesystemNode*> *node*)

void **resetItems** (*sh::filesystem::FilesystemNodeType* *type*, QSet<std::shared\_ptr<*FilesystemNode*>> *nodes*)

std::shared\_ptr<*FilesystemNode*> **myNode** ()

Returns the node which owns this children list. The result may be 0 for non model-backed lists.

void **addItem** (QSet<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> *nodes*) = 0

Adds items to this list. In a model-backed list, this triggers the internal model mounting calls. It is not allowed to call this method twice with the same node.

```

void addItem (std::shared_ptr<sh::filesystem::FilesystemNode> node) = 0
    Adds an item to this list. In a model-backed list, this triggers the internal model mounting calls. It is
    not allowed to call this method twice with the same node.

void removeItems (QSet<std::shared_ptr<sh::filesystem::FilesystemNode>> nodes) = 0
    Removes nodes from this list. In a model-backed list, this triggers the internal model unmounting
    calls. It is not allowed to call this method with a node, which is not contained in that list.

void removeItem (std::shared_ptr<sh::filesystem::FilesystemNode> node) = 0
    Removes a node from this list. In a model-backed list, this triggers the internal model unmounting
    calls. It is not allowed to call this method with a node, which is not contained in that list.

void resetItems (sh::filesystem::FilesystemNodeType type, QSet<std::shared_ptr<sh::filesystem::FilesystemNode>>
    nodes) = 0
    Resets the list for a given node type to a given new list of children nodes.

bool contains (std::shared_ptr<FilesystemNode> node)
    Checks if this list contains a given node.

const QList<std::shared_ptr<sh::filesystem::FilesystemNode>> *nodes ()
    Returns the list of nodes currently stored in this instance.

class DetailColumn : public std::enable_shared_from_this<DetailColumn>
    #include <detailcolumn.h> Abstract base class for a detail column (on filesystem nodes).

    Those can e.g. be seen in the file list view, but can also be queried internally by other places in code.

    It encapsulates the retrieval logic and metadata for one piece of additional information a
    sh::filesystem::FilesystemNode can have (e.g. the filesize). Retrieving the values is designed to be asyn-
    chronous. Each instance represents one column, while the actual logic is implemented in subclasses. For
    a new detail column, subclass this class and implement at least determineValue.

    Subclassed by sh::detailcolumns::DetailColumnCustomAttributes, sh::detailcolumns::DetailColumnDirectSymlinkTarget,
    sh::detailcolumns::DetailColumnExtendedAttributes, sh::detailcolumns::DetailColumnFilesize,
    sh::detailcolumns::DetailColumnMimetype, sh::detailcolumns::DetailColumnMtime,
    sh::detailcolumns::DetailColumnResolvedSymlink, sh::filesystemhandlers::SharCFilesystemHandler::ArchivedSizeDetailCo
    sh::scripting::api::ApiDetailColumn

```

## Public Functions

QString **name** ()

The internal unique name.

QString **displayName** ()

The display name.

bool **isValueAvailable** (std::shared\_ptr<FilesystemNode> node)

Checks if a value for this detail is available for one given node.

QVariant **value** (std::shared\_ptr<FilesystemNode> node, bool ignoreAged = false)

Returns the value for this detail for one given node.

### Parameters

- ignoreAged: If no value should be returned which is older than the last request (not useful for nearly all cases).

QString **displayValue** (std::shared\_ptr<FilesystemNode> node, const sh::filesystem::FilesystemModelFileviewProxy \*viewmodel)

Returns the stringified value for this details for one given node considering the configuration of a view.

bool **isVisible** ()

If this detail shall be a visible column in the view.

QVariant **requestValue** (std::shared\_ptr<FilesystemNode> node, sh::filesystem::Operation \*op = 0, bool withNodeValues = false, bool withOperationsCache = true)

Requests to determine the value for this detail for one given node. Blocks until the value is available. Get it with the value method afterwards.

bool **sort\_doTypediff** ()

If sorting should separate files and directories.

int **sort\_order** (std::shared\_ptr<FilesystemNode> n1, std::shared\_ptr<FilesystemNode> n2)

The sorting order.

uint **displayIndex** ()

Controls which place this column should get in the user interface.

QVariant **computeValue** (std::shared\_ptr<FilesystemNode> node, sh::filesystem::Operation \*op)

Returns a freshly determined value for this column and the given node. It neither asks nor populates any caches or storages.

int **defaultWidth** ()

bool **isRightAligned** ()

~DetailColumn ()

void **applyValueByEurl** (std::shared\_ptr<const sh::filesystem::Eurl> eurl, sh::filesystem::Operation \*op, QVariant value)

Applies the detail value to a given eurl (without using any caches).

Used for transferring some details when a file transfer occurs.

Optionally implement the one of the applyValueBy... methods and register the instance with sh::filesystem::FilesystemOperation::addTransferrableDetailColumn. For performance reasons, you should decide to implement applyValueByEurl.

void **applyValueByNode** (std::shared\_ptr<FilesystemNode> node, sh::filesystem::Operation \*op, QVariant value)

Applies the detail value to a given node (without using any caches).

Used for transferring some details when a file transfer occurs.

Optionally implement the one of the applyValueBy... methods and register the instance with sh::filesystem::FilesystemOperation::addTransferrableDetailColumn. For performance reasons, you should decide to implement applyValueByEurl.

## Public Static Functions

QVariant **VALUE\_UNAVAILABLE** ()

A special value expressing that the value is not yet available.

void **registerKnownDetailColumn** (QString name, std::shared\_ptr<DetailColumn> column)

std::shared\_ptr<DetailColumn> **findKnownDetailColumn** (QString name)

### Public Static Attributes

```

const uint DISPLAYINDEX_CORE = 10000
const uint DISPLAYINDEX_INTERESTING = 20000
const uint DISPLAYINDEX_EXOTIC = 30000

```

### Private Members

```

QString _displayName
QString _name
uint _displayIndex
bool _sort_doTypediff
int _defaultWidth
bool _isRightAligned

```

### Private Static Attributes

```

QMap<QString, std::shared_ptr<DetailColumn>> _knownDetailColumns
QMutex _knownDetailColumnsMutex

```

### Friends

```

friend class sh::filesystem::FilesystemNode

class Eurl : public std::enable_shared_from_this<Eurl>
#include <eurl.h> Filesystem paths.

```

A Extended Uniform Resource Locator is something like a [URL](#).

It is more general since it can also be nested. It is something like `zip:/{}/zippedfolder/zippedfile`.

You get [Eurl](#) instances only from the factory methods. There will never be two instances with the same textual value.

Please note: Instances can be associated with any kind of elements in the filesystem (files, directories, links, ...). It might also point to something which does not exist at all. The documentation sometimes explicitly makes a difference between those kinds (files, directories, links, ...; often called ‘node type’). But it often uses the term ‘file’ implicitly while meaning all kinds of elements; assuming that e.g. a directory is just a special kind of a file. It should be clear from the particular context which meaning applies.

## Public Functions

### `QString asString () const`

Returns the textual value. This is what *Eurl.fromString* would expect as parameter.

### `QString hostname () const`

Returns the hostname part (from the outer url of this *Eurl*). Examples: "livingroom-pc" for `smb://livingroom-pc/foo/bar/baz`. "" for `.`

### `QString path () const`

Returns the path part (from the outer url of this *Eurl*). Examples: `"/foo/bar/baz"` for `smb://livingroom-pc/foo/bar/baz`. `"/foo/bar/baz"` for `.` `"/"` for `.`

### `QString basename () const`

Returns the last path segment. This is the text behind the last slash. Examples: "baz" for `.` "" for `.`

### `QString scheme () const`

Returns the scheme (from the outer url of this *shallot.Eurl*). This is what comes before the `://`. Example: "file" for `.`

### `std::shared_ptr<const Eurl> outerUrl () const`

Returns a new *Eurl* containing only the outer part of this one (strips the embeddings). Example: `foobar://host/foo/bar/baz` for `foobar:[zip://d/e]/foo/bar/baz`.

### `std::shared_ptr<const Eurl> outermostInnerEurl () const`

Returns a new *Eurl* containing only the embedding of this one. Example: `zip://d/e` for `foobar:[zip://d/e]/foo/bar/baz`.

### `std::shared_ptr<const Eurl> withAppendedSegment (QString basename) const`

Returns a new *Eurl* from this one with `"/basename"` appended. The parameter is assumed to be a single path segment.

### `std::shared_ptr<const Eurl> withAppendedSegments (QString path) const`

Returns a new *Eurl* with path segments `"/pa/t/h/"` appended. The parameter may contain `"/"`s for dividing path segments.

### `std::shared_ptr<const Eurl> root () const`

Returns the root *Eurl* from this one. Example: `zip://` for `zip://foo/bar/baz`.

### `std::shared_ptr<const Eurl> unwrapWithOuterUrl (QString scheme, QString hostname, QString path) const`

Returns a new *Eurl* containing this one packed as embedding and new outer parts scheme, hostname and path. Example: `scheme://hostname/p/a/t/h` for `.`

### `std::shared_ptr<const Eurl> parentSegment () const`

Returns the parent *Eurl*. At first, this traverses path segments. For a root path eurl with embeddings, it returns the embedding. If none are available, it returns 0. Examples: `foo://host/foo` for `foo://host/foo/bar`. `foo://host/` for `foo://host/boo`. `foo:[bar://host/goo]/anotherhost/` for `foo:[bar://host/goo]/anotherhost/boo`. `bar://host/foo` for `foo:[bar://host/foo]/host/`. 0 for `foo://host/`.

### `bool hasInnerUrls () const`

Checks if this *Eurl* has embeddings. Examples: `true` for `foo:[bar:///foo]/host/`. `false` for `foo://host/`.

### `bool outerUrlIsRootDirectory () const`

Checks if this *Eurl* is a root path (with or without embeddings). Examples: `true` for `foo://host/`. `false` for `foo://host/a`. `true` for `foo:[bar:///goo]/host/`. `false` for `foo:[bar:///goo]/host/a`. `true` for `foo:[bar:///goo]/`. `false` for `foo:[bar:///goo]/a`.



**bool hasParentSegment () const**

Checks if this *Eurl* has a parent segment. This indicates if *Eurl.parentSegment* would return 0.

**bool isPrefixOf (std::shared\_ptr<const *Eurl*> longer) const**

Checks if this *Eurl* is a prefix of another one. This is not an equivalent to a string comparison but it checks parent relationships according to *Eurl.parentSegment*.

**Eurl (QString *eurlstring*)**

Constructed only by the infrastructure and made available otherwise.

**~Eurl ()**

## Public Static Functions

**std::shared\_ptr<const *Eurl*> fromString (QString *eurlstring*)**

Constructs a new *Eurl* by string (what *Eurl.asString* would return).

**std::shared\_ptr<const *Eurl*> create (QString *scheme*, QString *hostname*, QString *path*)**

Constructs a new *Eurl* by scheme name, hostname and a path.

**std::shared\_ptr<const *Eurl*> create (QString *scheme*, const *Eurl* \**inner*, QString *hostname*, QString *path*)**

Constructs a new *Eurl* by scheme name, an inner *eurl*, a hostname and a path.

**void filenameCheck (QString *filename*)**

Checks if a name is a valid filename. If not, *EurlMisformattedException* is thrown.

**void doInitialize ()**

**void doShutdown ()**

## Public Static Attributes

**const QChar WRAPPER\_BEGIN = '['**

The character marking the begin of an embedding.

**const QChar WRAPPER\_END = ']'**

The character marking the end of an embedding.

**const QString FORBIDDEN\_FILENAME\_CHARACTERS = QString("/")**

Characters which are forbidden in filenames.

## Private Members

**const QString \_eurlstring**

**std::shared\_ptr<const *Eurl*> \_cache\_parentsegment = 0**

**std::shared\_ptr<const *Eurl*> \_cache\_outerurl = 0**

**bool \_cache\_outerurl\_isthis = false**

**std::shared\_ptr<const *Eurl*> \_cache\_outermostinnereurl = 0**

**std::shared\_ptr<const *Eurl*> \_cache\_root = 0**

**bool \_cache\_root\_isthis = false**

**QString \_cache\_basename**

**QString \_cache\_hostname**

QString **\_cache\_path**  
QString **\_cache\_scheme**

### Private Static Functions

QString **check\_scheme** (QString *scheme*)  
QString **check\_inner** (QString *inner*)  
QString **check\_hostname** (QString *hostname*)  
QString **check\_path** (QString *path*)  
QString **check\_filename** (QString *filename*)  
QString **\_escape** (QString *s*)  
QString **\_unescape** (QString *s*)  
std::shared\_ptr<const *Eurl*> **createNOCHECK** (QString *scheme*, const *Eurl* \**inner*, QString  
*hostname*, QString *path*)  
std::shared\_ptr<const *Eurl*> **fromStringNOCHECK** (QString *eurlstring*)

### Private Static Attributes

QHash<QChar, QString> **\_escapemap**  
QMutex **\_escapemapmutex**  
QMutex **\_mutex**  
QHash<QString, std::weak\_ptr<const *Eurl*>> **\_eurluniverse**  
**class EurlMisformattedException** : public *sh::exceptions::ArgumentException*  
*#include <eurl.h>* Shallot exception for invalid input in *Eurl* creation methods.

### Public Functions

**EurlMisformattedException** (*sh::exceptions::ExceptionData* *data*)  
QString **message** () const  
QString **name** () const  
QString **classes** () const  
QString **details** () const  
QString **callstack** () const  
QString **auxiliary** () const  
QString **customValue** (QString *key*) const  
bool **isRuntimeException** () const  
bool **isProgramException** () const  
bool **isRetryable** () const  
bool **isResumeable** () const

```

bool isDetailsAreInteresting () const
int autoRetryRecommendedIn () const
void setResumeable (bool v)
void setReTryable (bool v)
void setCustomValue (QString key, QString value)
bool isClass (QString classname)
sh::exceptions::ExceptionData data ()

```

## Public Static Functions

```

template<class Handler>
std::shared_ptr<RegisterHandler<Handler>> createRegisterHandler (Handler handler,
                                                                QStack<Handler>
                                                                *stack)

void executeGuarded (std::function<void>)
    > fctint flags = 0 Executes some code with some standard exection handling around.

    Parameters
    • fct: The code to execute guarded.
    • flags: Flags of ExecuteGuardFlag for choosing the behavior.

void executeGuarded_errorpanel (std::function<void>)
    > fctint flags = 0

QString _demangle (QString l)

void exceptionDialog (sh::exceptions::Exception &e, bool *doRetry)

```

## Public Static Attributes

```

const QString UNDEFINED_ERROR_OCCURRED = QObject::tr("An unspecified error occurred.")
const QString SHALLOT_MUST_CLOSE_TEXT = QObject::tr("Your Shallot process is disturbed by an error and needs to be restarted.")
const QString Value_isShallotException = "_isShallotException"

class FilesystemHandler : public QObject
    #include <filesystemhandler.h> Abstract base class for a custom filesystem handler.

    Subclass and register it for implementing a new virtual filesystem, which lets new nodes appear somewhere
    in the filesystem tree and controls how to handle them.

    Use sh::filesystem::FilesystemHandlerRegister for registration.

    For executing some actions or checks on a filesystem, you should not use those handlers directly, but the
    higher-level sh::filesystem::FilesystemOperation class.

    Subclassed by sh::filesystemhandlers::ArchiveFilesystemHandler, sh::filesystemhandlers::GnomeIOFilesystemHandler,
    sh::filesystemhandlers::LocalFilesystemHandler, sh::filesystemhandlers::SharcFilesystemHandler,
    sh::scripting::api::ApiFilesystemHandler, sh::search::SearchFilesystemHandler

```

## Public Functions

**FilesystemHandler** (*sh::filesystem::FilesystemModel \*model*)

Is (for subclasses) intended to be directly constructed and registered once.

**~FilesystemHandler** ()

void **itemlist** (*sh::filesystem::Operation \*op*, std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*,  
*sh::filesystem::FilesystemNodeType type*, *sh::filesystem::FilesystemNodeListEditor*  
*\*list*) = 0

Determine a list of subelements in a certain directory.

void **configureItems** (*sh::filesystem::Operation \*op*, QList<std::shared\_ptr<*FilesystemNode*>>  
*nodes*)

Configure newly created nodes (e.g. setting another icon or changing the display name).

*sh::filesystem::FilesystemNodeType* **getType** (*sh::filesystem::Operation* *\*op*,  
std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*) =  
0

Determine if a file is regular, or a dir, or a link, ...

qint64 **getSize** (*sh::filesystem::Operation \*op*, std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*)  
= 0

Determine the size of a file.

QDateTime **getMtime** (*sh::filesystem::Operation \*op*, std::shared\_ptr<const *sh::filesystem::Eurl*>  
*eurl*) = 0

Determine the mtime of a file.

void **setMtime** (*sh::filesystem::Operation \*op*, std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*,  
QDateTime *time*) = 0

Set the mtime of a file.

QString **getMimetype** (*sh::filesystem::Operation* *\*op*, std::shared\_ptr<const  
*sh::filesystem::Eurl*> *eurl*) = 0

Determine mime type of a file.

QString **getLinkTarget** (*sh::filesystem::Operation* *\*op*, std::shared\_ptr<const  
*sh::filesystem::Eurl*> *eurl*) = 0

Resolve a link.

QList<QString> **listExtendedAttributes** (*sh::filesystem::Operation* *\*op*,  
std::shared\_ptr<const *sh::filesystem::Eurl*>  
*eurl*)

Fetches the list of available extended attributes for an entry.

qint64 **getExtendedAttributeSize** (*sh::filesystem::Operation \*op*, std::shared\_ptr<const  
*sh::filesystem::Eurl*> *eurl*, QString *attribute*)

Returns the size of value for one extended attribute for an entry.

QByteArray **getExtendedAttribute** (*sh::filesystem::Operation \*op*, std::shared\_ptr<const  
*sh::filesystem::Eurl*> *eurl*, QString *attribute*)

Returns the value for one extended attribute for an entry.

void **setExtendedAttribute** (*sh::filesystem::Operation* *\*op*, std::shared\_ptr<const  
*sh::filesystem::Eurl*> *eurl*, QString *attribute*, QByteArray  
*value*)

Sets the value for one extended attribute for an entry.

void **removeExtendedAttribute** (*sh::filesystem::Operation \*op*, std::shared\_ptr<const  
*sh::filesystem::Eurl*> *eurl*, QString *attribute*)

Removes one extended attributes for an entry.

QMap<QString, QString> **getCustomAttributes** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)

Returns the custom attributes for an entry.

In contrast to extended attributes, the custom attributes are not directly stored in the filesystem this way, but handle implementation specific aspects (like permissions).

void **setCustomAttribute** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QString attribute, QString value)

Sets the value for one custom attribute for an entry.

See also getCustomAttributes.

bool **canGetFileContent** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0

Returns if it is allowed to get the content of a certain file.

std::shared\_ptr<QIODevice> **getFileContent** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0

Get the content of a file.

bool **canCreateDirectory** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0

Returns if it is allowed to create subdirectories in a certain directory.

void **createDirectory** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0

Create a directory.

bool **canCreateLink** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0

Returns if it is allowed to create a link in a certain directory.

void **createLink** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QString target) = 0

Create a link.

bool **canCreateFile** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0

Returns if it is allowed to create files in a certain directory.

void **createFile** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QIODevice \*content, *HandlerTransfer* \*handlertransfer) = 0

Creates a file with some content.

It may use handlertransfer for some better integration, like cancel support and progress monitoring.

bool **canDeleteItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0

Returns if it is allowed to delete a certain file/directory/link/...

void **deleteItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0

Delete a file/directory/link/...

void **deleteDirectoryIfEmpty** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)

Deletes a directory only if it is empty.

bool **canRenameItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> src) = 0

Returns if it is allowed to rename a certain file, directory, link, ... (see renameItem).

void **renameItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> src, QString destpath) = 0  
Renames an item to another path.

It can be a simple filename change or also changes in the deeper path segments.

QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> **getActions** (const QList<std::shared\_ptr<const *sh::filesystem::Eurl*>> eurls) = 0

Returns a list of actions (see former example) for showing for certain files.

bool **requiresResolvedLinks** ()  
Returns if this handler requires to be used by the engine with resolved links.

void **viewEnteredDirectory** (std::shared\_ptr<const *sh::filesystem::Eurl*>)  
Callback for preparing stuff when the view entered some directory (e.g. for enabling watchers). See also viewLeftDirectory.

void **viewLeftDirectory** (std::shared\_ptr<const *sh::filesystem::Eurl*>)  
Callback for preparing stuff when the view left some directory. See also viewEnteredDirectory.

bool **isWellKnownScheme** ()  
Returns if the scheme for this handler is a well known one (which external programs typically would understand).

void **search** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, std::function<void> std::shared\_ptr<const *sh::filesystem::Eurl*>, QList<std::shared\_ptr<*sh::search::SearchCriterion*>>, *sh::filesystem::FilesystemNodeType* ftype  
> addfile, std::function<void>std::shared\_ptr<const *sh::filesystem::Eurl*>> visitdir, QList<std::shared\_ptr<*sh::search::SearchCriterion*>> criteriaHelps searching for files.

Override this method with a behavior identical to the default, if a specialized implementation can be much faster than the default one.

void **customizeUi** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> node, bool \*showSearchPanel)  
Creates a custom gui, which becomes part of the fileview.

*sh::filesystem::FilesystemModel* \***model** ()  
A convenience shortcut to the *sh::filesystem::FilesystemModel* (a singleton).

**class HandlerTransfer**  
#include <filesystemhandler.h> Subclassed by *sh::filesystem::FilesystemOperation::MyHandlerTransfer*,  
*sh::filesystem::FilesystemOperationTransfers::SingleStepMonitor*

## Public Functions

void **respectCancel** ()  
void **incrementTransferredBytes** (qint64 donebytes)  
~**HandlerTransfer** ()

**class FilesystemHandlerRegister** : public QObject, public *sh::base::Singleton*  
#include <filesystemhandlerregister.h> A register of filesystem handlers.

Each active (i.e. referred to by existing nodes) instance of *sh::filesystem::FilesystemHandler* must be registered here.

## Public Functions

void **addHandler** (QString *scheme*, std::shared\_ptr<sh::filesystem::FileSystemHandler> *handler*)  
Registers a filesystem handler.

### Parameters

- *scheme*: The scheme (very first part of a *sh::filesystem::Eurl*) for which the handler is responsible for.
- *handler*: The filesystem handler.

std::shared\_ptr<sh::filesystem::FileSystemHandler> **findHandler** (QString *scheme*)  
Finds a filesystem handler by scheme.

void **doInitialize** ()  
Executes singleton initialization.

void **doShutdown** ()  
Executes singleton shutdown.

void **shutdown** ()  
Shutdown down this singleton.

bool **isAlive** ()  
Returns if this singleton is alive (true until its shutdown begins).

## Private Functions

**FileSystemHandlerRegister** ()

## Private Members

QHash<QString, std::shared\_ptr<sh::filesystem::FileSystemHandler>> **handlers**

QMutex **mutex**

**class FileSystemModel** : public QAbstractItemModel, public sh::base::Singleton  
*#include <filesystemmodel.h>* The filesystem model.

This is the engine which creates and manages *sh::filesystem::FileSystemNode* nodes. Filesystem nodes are used on many places for all kinds of operations.

## Public Functions

std::shared\_ptr<sh::filesystem::FileSystemNode> **rootNode** ()  
The *sh::filesystem::FileSystemNode* which is the root node of the entire model. It is the parent for all toplevel nodes.

QVariant **data** (const QModelIndex &*index*, int *role* = Qt::DisplayRole) const

Qt::ItemFlags **flags** (const QModelIndex &*index*) const

QVariant **headerData** (int *section*, Qt::Orientation *orientation*, int *role* = Qt::DisplayRole) const

QModelIndex **index** (int *row*, int *column*, const QModelIndex &*parent* = QModelIndex()) const

QModelIndex **parent** (const QModelIndex &*index*) const

int **rowCount** (const QModelIndex &*parent* = QModelIndex()) const



int **columnCount** (const QModelIndex &parent = QModelIndex()) const

*sh::filesystem::FilesystemModelFileviewProxy* \***createFileviewProxy** (QModelIndex root, bool withtooltip, std::function<void> std::shared\_ptr<*sh::filesystem::FilesystemModelFileviewProxy*> proxy)  
*> onBecomesInvalid* Creates a *sh::filesystem::FilesystemModelFileviewProxy* for presenting the content of a directory.

QList<QPersistentModelIndex> **findIndexesForEurl** (std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)

Get a list of qt model indexes for a *sh::filesystem::Eurl*.

If nodes for this eurl are unknown to the model so far, it tries to build them. In typical cases, this list either contains one element, or is empty if the filesystem handlers decide that this file does not exist. But in some cases, there is also more than one index for one *sh::filesystem::Eurl* (when the *sh::filesystem::Eurl* appears on more than one place in the tree).

QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> **findNodesForEurl** (std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)

Get a list of *sh::filesystem::FilesystemNode* for a *sh::filesystem::Eurl*.

If it is unknown to the model so far, it tries to build them. In typical cases, this list either contains one element, or is empty if the filesystem handlers decide that this file does not exist. But in some cases, there is also more than one node for one *sh::filesystem::Eurl* (when the *sh::filesystem::Eurl* appears on more than one place in the tree). It only returns nodes, which are 'alive', i.e. which have a living parent and which are a child of this parent.

QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> **tryGetNodesForEurl** (std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)

Returns a list of *sh::filesystem::FilesystemNode* for a *sh::filesystem::Eurl*. It only considers the current state of the in-memory model. It will only return nodes which are already known to the model so far. This operation is cheaper and handling only the known nodes is sufficient in many situations. In typical cases, this list either contains one element, or is empty. But in some cases, there is also more than one node for one *sh::filesystem::Eurl* (when the *sh::filesystem::Eurl* appears on more than one place in the tree). It only returns nodes, which are 'alive', i.e. which have a living parent and which are a child of this parent.

std::shared\_ptr<*sh::filesystem::FilesystemNode*> **getNodeForIndex** (const QModelIndex index) const

Returns the *sh::filesystem::FilesystemNode* for a qt model index (in the main model).

QModelIndex **getIndexForNode** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> node)

Returns a qt model index (in the main model) for a *sh::filesystem::FilesystemNode*.

std::shared\_ptr<*sh::filesystem::FilesystemNode*> **createFilesystemNode** (std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, *sh::filesystem::FilesystemHandler* \*handler, *sh::filesystem::FilesystemNodeType* nodetype, bool isHidden, std::shared\_ptr<*sh::filesystem::FilesystemNode*> parent, bool doinsert = true, bool showInitialLoadingLabel = true)

Creates a new *sh::filesystem::FilesystemNode* for placing it into the model.



If such a node (with the same eurl for the same parent node) does not exist, it generates a new one. If there already is such a node alive, but currently not placed in the model, it recycles this one. This can happen when references exist to a node which is not yet inserted or which is removed meanwhile. It is not allowed to call this method when such a node already exists in the model.

Use this function for getting a *sh::filesystem::FilesystemNode*, which is to be added to the model now or later. Depending on some parameter values, a call directly adds the node to the filesystem model (not e.g. when `doinsert` is `false` or `parentnode` is 0) It is typically used within a *sh::filesystem::FilesystemHandler* implementation.

```
std::shared_ptr<sh::filesystem::FilesystemNode> getOrCreateFilesystemNode (std::shared_ptr<const
sh::filesystem::Eurl>
neweurl,
sh::filesystem::FilesystemHandler
*handler,
sh::filesystem::FilesystemNodeType
node-
type, bool
isHidden,
std::shared_ptr<sh::filesystem::Files
parent, bool
doinsert =
true, bool
showIni-
tialLoad-
ingLabel =
true, bool
*plIsNew =
0)
```

Returns a *sh::filesystem::FilesystemNode* for using it as a child node in the model.

It either creates a new one, if there currently is no node for this eurl in this parentnode, or returns the existing one. Even for existing ones, this call can change the nodetype of that node.

Depending on some parameter values, a call directly adds the node to the filesystem model (not e.g. when `doinsert` is `false` or `parent` is 0).

```
void refreshData (std::shared_ptr<const sh::filesystem::Eurl> eurl, bool forceFindParent =
false, bool withDetails = true)
```

Request to refresh the internal model information for a *sh::filesystem::Eurl*. This may be a place which is already known (then a change of some metadata or the removal is detected) or a formerly unknown place (then new nodes get inserted in the model).

```
void addOpenNodeHelper (int index, std::function<QList<std::shared_ptr<sh::filesystem::FilesystemNode>>>() std::share
sh::filesystem::Eurl>
> openNodeHelperRegisters a helper method for ‘opening’ (mounting, activating, ...) locations on
demand.
```

Only used in very rare cases.

```
~FilesystemModel ()
```

```
void doInitialize ()
```

Executes singleton initialization.

```
void doShutdown ()
```

Executes singleton shutdown.

```
void shutdown ()
```

Shutdown down this singleton.

bool **isAlive** ()  
Returns if this singleton is alive (true until its shutdown begins).

### Private Functions

**FilesystemModel** ()

QList<std::shared\_ptr<sh::filesystem::FilesystemNode>> **\_findNodesForEurl\_helper** (std::shared\_ptr<const sh::filesystem::Eurl> eurl)

QList<std::shared\_ptr<sh::filesystem::FilesystemNode>> **openRootEurl** (std::shared\_ptr<const sh::filesystem::Eurl> eurl)

void **\_subitemFetchingStateChanged** (std::shared\_ptr<sh::filesystem::FilesystemNode> node, bool value)

### Private Members

QList< std::function< QList< std::shared\_ptr< sh::filesystem::FilesystemNode > >std::shared\_ptr<sh::filesystem::FilesystemNode> >>

QMap< int, std::function< QList< std::shared\_ptr< sh::filesystem::FilesystemNode > >std::shared\_ptr<sh::filesystem::FilesystemNode> >>

std::shared\_ptr<sh::filesystem::FilesystemNode> **rootnode**

QMutex **mutex**

QHash<std::shared\_ptr<const sh::filesystem::Eurl>, std::weak\_ptr<sh::filesystem::FilesystemNode>> **eurl2node**

QMutex **eurl2nodemutex**

QMutex **\_nodeDataMutex**

QMutex **\_openNodeHelpersMutex**

### Friends

**friend class** FilesystemNode

**friend class** LoadOnDemandPlaceholderFilesystemNode

**friend class** ModelBackedFilesystemNodeList

**friend class** ::sh::filesystem::DetailColumn

**class FilesystemModelDirectoryTreeProxy** : public QSortFilterProxyModel, public sh::base::Singleton  
#include <filesystemmodeldirectorytreeproxy.h> A filesystem proxy model for the directory tree.

It has a special sorting and filtering behavior.

Used internally, mostly for the user interface.

## Public Functions

void **enforceVisibility** (std::shared\_ptr<*FilesystemNode*> node)  
 Marks a node as visible in the directory tree, even if it is a hidden node.

void **deEnforceVisibility** (std::shared\_ptr<*FilesystemNode*> node)  
 Unmarks a node for being visible even if hidden (reverses *enforceVisibility*()).

void **doShutdown** ()  
 Executes singleton shutdown.

void **shutdown** ()  
 Shutdown down this singleton.

bool **isAlive** ()  
 Returns if this singleton is alive (true until its shutdown begins).

## Private Functions

**FilesystemModelDirectoryTreeProxy** ()

**class FilesystemModelDirectoryTreeProxyVisibilityEnforcements** : public QObject, public *sh::base::Singleton*  
*#include <filesystemmodeldirectorytreeproxyvisibilityenforcements.h>* Maintains a list of currently visible directories (i.e. the current one in each view) and controls enforced visibility of them in the directory tree.

## Public Functions

void **nodeEnteredView** (std::shared\_ptr<*FilesystemNode*> node)  
 Called when a view enters the given directory node.

void **nodeLeftView** (std::shared\_ptr<*FilesystemNode*> node)  
 Called when a view leaves the given directory node.

void **nodeCollapsedInTree** (std::shared\_ptr<*FilesystemNode*> node)  
 Called when a directory node is collapsed in the directory tree.

void **doShutdown** ()  
 Executes singleton shutdown.

void **shutdown** ()  
 Shutdown down this singleton.

bool **isAlive** ()  
 Returns if this singleton is alive (true until its shutdown begins).

## Private Functions

**FilesystemModelDirectoryTreeProxyVisibilityEnforcements** ()

bool **\_addenforcement** (std::shared\_ptr<*FilesystemNode*> node, std::shared\_ptr<*FilesystemNode*> hnode)

## Private Members

QMultiMap<std::shared\_ptr<*FilesystemNode*>, QObject\*> **\_enterednodesrepresentatives**

QList<std::tuple<std::weak\_ptr<*FilesystemNode*>, QList<std::shared\_ptr<*FilesystemNode*>>>> **hiddenForcedVisible**

**class FilesystemModelFileviewProxy : public** QSortFilterProxyModel  
*#include <filesystemmodelfileviewproxy.h>* A filesystem proxy model for a fileview.

It regards the sorting and filtering behavior set up for the connected fileview.

Used internally, mostly for the user interface.

## Public Functions

**FilesystemModelFileviewProxy** (QModelIndex *root*, bool *withtooltip*, QObject *\*parent* = 0)

Constructed only indirectly.

void **setSizeFormattingMode** (*SizeFormatting mode*)

*SizeFormatting* **getSizeFormattingMode** () **const**

void **setHiddenFilesVisible** (bool *v*)

bool **getHiddenFilesVisible** () **const**

QVariant **data** (const QModelIndex &*index*, int *role* = Qt::DisplayRole) **const**

void **triggerReloadData** ()

void **setThumbnail** (bool *enabled*, int *size* = 32)

bool **getThumbnailEnabled** ()

int **getThumbnailSize** ()

**class FilesystemModelSubtreeProxy : public** QAbstractItemModel  
*#include <filesystemmodelsubtreeproxy.h>* A filesystem proxy model for setting a new root node to an existing fileview proxy.

Used internally, mostly for the user interface.

## Public Functions

**FilesystemModelSubtreeProxy** (QModelIndex *root*, *sh::filesystem::FilesystemModelFileviewProxy* *\*upperproxy*)

Constructed only indirectly.

QVariant **data** (const QModelIndex &*index*, int *role*) **const**

Qt::ItemFlags **flags** (const QModelIndex &*index*) **const**

QVariant **headerData** (int *section*, Qt::Orientation *orientation*, int *role* = Qt::DisplayRole) **const**

QModelIndex **index** (int *row*, int *column*, const QModelIndex &*parent* = QModelIndex()) **const**

QModelIndex **parent** (const QModelIndex &*index*) **const**

int **rowCount** (const QModelIndex &*parent* = QModelIndex()) **const**

int **columnCount** (const QModelIndex &*parent* = QModelIndex()) **const**

QModelIndex **mapFromSource** (const QModelIndex *mi*) **const**

QModelIndex **mapToSource** (const QModelIndex *mi*) const

## Signals

void **becomesInvalid** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> *bestValid*)

## Private Members

bool **\_inChangeTransaction**

std::shared\_ptr<*sh::filesystem::FilesystemNode*> **\_rootnode**

QPersistentModelIndex **\_rootindex**

*sh::filesystem::FilesystemModel* \***mainmodel**

*sh::filesystem::FilesystemModelFileviewProxy* \***\_upperproxy**

**class FilesystemNode** : public QObject, public std::enable\_shared\_from\_this<*FilesystemNode*>  
*#include <filesystemnode.h>* A representation of a location in the filesystem tree.

It represents filesystem nodes like a file or a directory in *sh::filesystem::FilesystemModel*.

Please note: Instances can be associated with any kind of elements in the filesystem (files, directories, links, ...). It might also point to something which does not exist at all (see parentnode). The documentation sometimes explicitly makes a difference between those kinds (files, directories, links, ...; often called 'node type'). But it often uses the term 'file' implicitly while meaning all kinds of elements; assuming that e.g. a directory is just a special kind of a file. It should be clear from the particular context which meaning applies.

Subclassed by *sh::filesystem::LoadOnDemandPlaceholderFilesystemNode*

## Public Functions

**~FilesystemNode** ()

std::shared\_ptr<const *sh::filesystem::Eurl*> **eurl** ()

Returns the *sh::filesystem::Eurl* entry address.

*sh::filesystem::FilesystemModel* \***model** ()

Convenience shortcut to the *sh::filesystem::FilesystemModel*.

QString **displayName** ()

Returns the display name (what the user interface displays).

void **setDisplayName** (QString *displayname*)

Sets the display name (what the user interface displays).

*sh::filesystem::FilesystemHandler* \***handler** ()

Returns the handler which is responsible for this node. This should be the same as *sh::filesystem::FilesystemHandlerRegister.findHandler*(eurl.scheme).

int **nodetype** ()

Returns the FilesystemNodeType node type.

int **childnodeCount** ()

Returns the number of children nodes.

std::shared\_ptr<*sh::filesystem::FilesystemNode*> **childnode** (int *i*)

Returns i-th child node.

*sh::filesystem::ModelBackedFilesystemNodeList* \***childnodes** ()

Returns the list of children.

std::shared\_ptr<*sh::filesystem::FilesystemNode*> **parentnode** () **const**

Returns the parent node.

This is 0 if the node does not exist in the model (i.e. not yet inserted or removed afterwards). In most cases, this can be interpreted as ‘file currently does not exist’.

int **index** ()

Returns the index of this node in the parent’s list of children.

void **addChild** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> *node*)

Adds a new child. Not allowed to call more than once for a node.

void **removeChild** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> *node*)

Removes a new child. Not allowed to call more than once for a node.

bool **isFetchingSubitems** ()

Checks if currently subitems are fetched (i.e. the ‘loading’ node is shown).

QIcon **icon** ()

Returns the node icon.

void **setIcon** (QIcon *icon*)

Sets the node icon.

bool **isHidden** ()

Returns if the node is hidden.

Note: It’s not difficult for the user to also show the hidden items.

void **setHidden** (bool *v*)

Sets if the node is hidden.

See also *isHidden()*.

void **addDetail** (std::shared\_ptr<*sh::filesystem::DetailColumn*> *column*)

Adds a detail to this node.

std::shared\_ptr<*sh::filesystem::DetailColumn*> **getDetailColumnByIndex** (int *index*)

Returns the i-th detail column registered to this node.

int **getDetailColumnsCount** () **const**

Returns the number of detail columns registered to this node.

bool **isRootNode** () **const**

Checks if this is the root node of the model.

bool **isConfigured** () **const**

Checks if this node was already configured by a handler.

void **requestDetails** (bool *force* = true)

Requests to fetch details for this node.

bool **isAlive** () **const**

Checks if this node currently exists in the model.

QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> **linkDestinationNodes** () **const**

Returns the list of link source nodes. If this node is a link, the link destination nodes may influence the behavior and appearance of this node.

This information does not come from inside but must be set from outside the model implementation.

```
void setLinkDestinationNodes (QList<std::shared_ptr<sh::filesystem::FilesystemNode>>
                               linkdestinations)
    Sets the list of link destination nodes. If this node is a link, the link destination nodes may influence
    the behavior and appearance of this node.

void requestContainedItems (sh::filesystem::FilesystemNodeType type = FilesystemNode-
                           Type::NONE)
    Requests to fetch the children of this node with help of the filesystem handler.

FilesystemNode (sh::filesystem::FilesystemModel *model, std::shared_ptr<const
                 sh::filesystem::Eurl> eurl, QString displayname,
                 sh::filesystem::FilesystemHandler *handler, FilesystemNodeType nodetype,
                 bool ishidden)
    Constructed only by the infrastructure and made available otherwise.
```

## Signals

```
void removed ()
    Is emitted when this node is not placed in the tree anymore. This does not mean the physical deletion
    of the instance, but just means it is not alive from now on.

void _detailsAvailable ()
    Is emitted when values for detail columns arrived.

void iconChanged ()
    Is emitted when the icon changed.

void isHiddenChanged ()
    Is emitted when the hidden flag changed.
```

## Private Functions

```
void setDetail (std::shared_ptr<sh::filesystem::DetailColumn> column, QVariant value)
int getInsertPositionForDetailColumn (std::shared_ptr<sh::filesystem::DetailColumn>
                                         newCol)
```

## Friends

```
friend class sh::filesystem::FilesystemModel
friend class sh::filesystem::DetailColumn
friend class LoadOnDemandPlaceholderFilesystemNode
friend class sh::filesystem::FilesystemModelFileviewProxy
friend class sh::filesystem::FilesystemModelSubtreeProxy
friend class sh::filesystem::ModelBackedFilesystemNodeList

class FilesystemNodeList : public QObject
#include <filesystemnodelist.h> A data structure for children lists of filesystem nodes.

Different subclasses exist with different behaviors and use cases.

Subclassed by sh::filesystem::AdhocFilesystemNodeList, sh::filesystem::ModelBackedFilesystemNodeList
```

## Public Functions

void **addItem** (QSet<std::shared\_ptr<sh::filesystem::FilesystemNode>> nodes) = 0  
Adds items to this list. In a model-backed list, this triggers the internal model mounting calls. It is not allowed to call this method twice with the same node.

void **addItem** (std::shared\_ptr<sh::filesystem::FilesystemNode> node) = 0  
Adds an item to this list. In a model-backed list, this triggers the internal model mounting calls. It is not allowed to call this method twice with the same node.

void **removeItems** (QSet<std::shared\_ptr<sh::filesystem::FilesystemNode>> nodes) = 0  
Removes nodes from this list. In a model-backed list, this triggers the internal model unmounting calls. It is not allowed to call this method with a node, which is not contained in that list.

void **removeItem** (std::shared\_ptr<sh::filesystem::FilesystemNode> node) = 0  
Removes a node from this list. In a model-backed list, this triggers the internal model unmounting calls. It is not allowed to call this method with a node, which is not contained in that list.

void **resetItems** (sh::filesystem::FilesystemNodeType type, QSet<std::shared\_ptr<sh::filesystem::FilesystemNode>> nodes) = 0  
Resets the list for a given node type to a given new list of children nodes.

std::shared\_ptr<sh::filesystem::FilesystemNode> **myNode** () = 0  
Returns the node which owns this children list. The result may be 0 for non model-backed lists.

bool **contains** (std::shared\_ptr<FilesystemNode> node)  
Checks if this list contains a given node.

const QList<std::shared\_ptr<sh::filesystem::FilesystemNode>> **\*nodes** ()  
Returns the list of nodes currently stored in this instance.

## class FilesystemNodeListEditor

#include <filesystemodelist.h> A list editor for easily modifying a *FilesystemNodeList*.

Mainly used for listing child nodes in *sh::filesystem::FilesystemHandler.itemlist*. In most cases, you should just use *setItems*.

## Public Functions

**FilesystemNodeListEditor** (FilesystemNodeList \*list, std::shared\_ptr<const sh::filesystem::Eurl> parenteurl, sh::filesystem::FilesystemNodeType nodetype)

Constructed only by the infrastructure and made available otherwise.

**~FilesystemNodeListEditor** ()

void **setItems** (QList<QString> items)

Sets a new list content. This function is all you need in most situations. For iteratively adding nodes, which makes the intermediate results visible in the user interface, see *addItem*.

void **beginIterativeAdding** ()

Must be called before you begin to iteratively fill the children list (e.g. with *addItem*). You must also call *endIterativeAdding* afterwards.

void **addItem** (QString item)

Adds a new children to the list, directly showing that in the user interface, while you can proceed filling the list. Please read *beginIterativeAdding* as well! In most cases, you don't need this function.



```
std::shared_ptr<FilesystemNode> addCustomItem (std::shared_ptr<const sh::filesystem::Eurl>
                                              eurl, QString displayname = QString(),
                                              sh::filesystem::FilesystemHandler *handler =
                                              0)
```

Adds a new children to the list, directly showing that in the user interface, while you can proceed filling the list. Please read beginIterativeAdding as well! In most cases, you don't need this function.

Note: If you specify a handler, it must be the one matching to eurl's scheme.

```
void addExistingNodeItem (std::shared_ptr<sh::filesystem::FilesystemNode> node)
```

Adds a new children to the list, directly showing that in the user interface, while you can proceed filling the list. Please read beginIterativeAdding as well! In most cases, you don't need this function.

```
void endIterativeAdding ()
```

Must be called after you iteratively filled the children list with addItem. This automatically removes all the old nodes, which you haven't added in this session.

```
void setItemsAreHosts ()
```

After this call, the addItem method will consider the given names as hostnames instead of new path segments. The resulting *sh::filesystem::Eurl* will differ accordingly. This only makes sense as children for root eurls.

```
FilesystemNodeList *rawlist ()
```

Returns the *FilesystemNodeList* backend. You should rarely need it.

## Private Functions

```
std::shared_ptr<const sh::filesystem::Eurl> getChildEurl (std::shared_ptr<const
                                                         sh::filesystem::Eurl> eurl, QString
                                                         item)
```

## Private Members

```
FilesystemNodeList *_list
```

```
std::shared_ptr<const sh::filesystem::Eurl> _parenteurl
```

```
sh::filesystem::FilesystemNodeType _nodetype
```

```
std::shared_ptr<sh::filesystem::FilesystemNode> _parentnode
```

```
sh::filesystem::FilesystemModel *_model
```

```
sh::filesystem::FilesystemHandler *_handler
```

```
bool _itemsAreHosts = false
```

```
bool _beganAddingIteratively = false
```

```
QSet<std::shared_ptr<sh::filesystem::FilesystemNode>> _iterativeAddingsBeforeState
```

### Private Static Attributes

QSet<FileSystemNodeList\*> **\_pendingIterativeAddings**

QMutex **\_pendingMutex**

QWaitCondition **\_iterativeAddingPossibleCondition**

### class FileSystemOperation

*#include <filesystemoperation.h>* A high-level interface for filesystem operations.

It is always based on the transaction of a *Operation* instance (which also gives you access to a *FileSystemOperation* object).

Some calls optionally allow to give an own instance of *sh::filesystem::FileSystemOperationProgressMonitor* for some additional functionality. Please note that not all calls provide all those functionality (some do not use e.g. the conflict resolution at all). If not provided, a default behavior is applied.

### Public Functions

**FileSystemOperation** (*sh::filesystem::Operation* \*operation)

Constructed only by the infrastructure and made available otherwise.

QList<std::shared\_ptr<const *sh::filesystem::Eurl*>> **itemlist** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*,  
*sh::filesystem::FileSystemNodeType*  
*type* = *sh::filesystem::FileSystemNodeType::NONE*)

Gets a list of shallot.sh::filesystem::FileSystemNode nodes in a directory (optionally filter by given types).

*sh::filesystem::FileSystemNodeType* **getType** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*)

Gets the shallot.sh::filesystem::FileSystemNodeType node type for an entry.

qint64 **getFileSize** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*)

Gets the file size for an entry.

QString **getLinkTarget** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*)

Gets the link target for an entry (if it is a link).

bool **canGetFileContent** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*)

Can we get the file content for an entry?

std::shared\_ptr<QIODevice> **getFileContent** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*)

Gets the file content for an entry as QIODevice.

bool **canCreateDirectory** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*)

Can we create a given directory?

void **createDirectory** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*,  
std::shared\_ptr<*sh::filesystem::FileSystemOperationProgressMonitor*>  
*progressmon* = 0)

Creates a directory.

#### Parameters

- *progressmon*: An optional progress monitor.

bool **canCreateLink** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*)

Can we create a given link?

```
void createLink (std::shared_ptr<const sh::filesystem::Eurl> eurl, QString target,
                std::shared_ptr<sh::filesystem::FilesystemOperationProgressMonitor> pro-
                gressmon = 0)
```

Creates a link.

#### Parameters

- *progressmon*: An optional progress monitor.

```
bool canCreateFile (std::shared_ptr<const sh::filesystem::Eurl> eurl)
    Can we create a given file?
```

```
void createFile (std::shared_ptr<const sh::filesystem::Eurl> eurl, QIODevice *content,
                std::shared_ptr<sh::filesystem::FilesystemOperationProgressMonitor> pro-
                gressmon = 0)
```

Creates a file.

#### Parameters

- *progressmon*: An optional progress monitor.

```
bool canDeleteItem (std::shared_ptr<const sh::filesystem::Eurl> eurl)
    Can we delete a given entry?
```

```
void deleteItem (std::shared_ptr<const sh::filesystem::Eurl> eurl,
                std::shared_ptr<sh::filesystem::FilesystemOperationProgressMonitor> pro-
                gressmon = 0)
```

Delete an entry.

#### Parameters

- *progressmon*: An optional progress monitor.

```
void deleteDirectoryIfEmpty (std::shared_ptr<const sh::filesystem::Eurl> eurl,
                             std::shared_ptr<sh::filesystem::FilesystemOperationProgressMonitor>
                             progressmon = 0)
```

Delete a directory entry if empty.

#### Parameters

- *progressmon*: An optional progress monitor.

```
bool canMoveItem (std::shared_ptr<const sh::filesystem::Eurl> src, std::shared_ptr<const
                  sh::filesystem::Eurl> dest = 0)
    Checks if it is allowed to move a certain item.
```

If the destination is known as well, it might help to pass it as well.

This does not guarantee success in the transfer, but is just a cheap early check.

```
void moveItems (QList<std::shared_ptr<const sh::filesystem::Eurl>>
                src, std::shared_ptr<const sh::filesystem::Eurl> dest,
                std::shared_ptr<sh::filesystem::FilesystemOperationProgressMonitor> pro-
                gressmon = 0)
```

Moves entries.

#### Parameters

- *dest*: The requested new common parent directory.
- *progressmon*: An optional progress monitor.

```
void moveItem (std::shared_ptr<const sh::filesystem::Eurl> src,
               std::shared_ptr<const sh::filesystem::Eurl> dest,
               std::shared_ptr<sh::filesystem::FilesystemOperationProgressMonitor> pro-
               gressmon = 0)
```

Moves an entry.

#### Parameters

- `dest`: The new location (not the new parent).
- `progressmon`: An optional progress monitor.

bool **canCopyItem** (std::shared\_ptr<const *sh::filesystem::Eurl*> *src*, std::shared\_ptr<const *Eurl*> *dest* = 0)

Checks if it is allowed to copy a certain item.

If the destination is known as well, it might help to pass it as well.

This does not guarantee success in the transfer, but is just a cheap early check.

void **copyItems** (QList<std::shared\_ptr<const *sh::filesystem::Eurl*>>  
*src*, std::shared\_ptr<const *sh::filesystem::Eurl*> *dest*,  
std::shared\_ptr<*sh::filesystem::FilesystemOperationProgressMonitor*> *pro-*  
*gressmon* = 0)

Copies entries.

#### Parameters

- `dest`: The requested new common parent directory.
- `progressmon`: An optional progress monitor.

void **copyItem** (std::shared\_ptr<const *sh::filesystem::Eurl*> *src*,  
std::shared\_ptr<const *sh::filesystem::Eurl*> *dest*,  
std::shared\_ptr<*sh::filesystem::FilesystemOperationProgressMonitor*> *pro-*  
*gressmon* = 0)

Copies an entry.

#### Parameters

- `dest`: The new location (not the new parent).
- `progressmon`: An optional progress monitor.

QList<std::shared\_ptr<*FilesystemNode*>> **resolveNodeLink** (std::shared\_ptr<*FilesystemNode*>  
*node*, bool *recursive* = true, bool  
*excludeOwn* = false)

Resolve a link as node with or without recursion.

std::shared\_ptr<const *sh::filesystem::Eurl*> **resolveEurlLink** (std::shared\_ptr<const  
*sh::filesystem::Eurl*> *eurl*,  
bool *recursive* = true, bool  
*excludeOwn* = false, int *tries* =  
100)

Resolve a link as *sh::filesystem::Eurl* with or without recursion.

QList<QString> **listExtendedAttributes** (std::shared\_ptr<const *sh::filesystem::Eurl*>  
*eurl*)

quint64 **getExtendedAttributeSize** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*,  
QString *attribute*)

QByteArray **getExtendedAttribute** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*,  
QString *attribute*)

void **setExtendedAttribute** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*, QString *at-*  
*tribute*, QByteArray *value*)

void **removeExtendedAttribute** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*, QString  
*attribute*)

## Public Static Functions

void **addTransferrableDetailColumn** (int *index*, std::shared\_ptr<sh::filesystem::DetailColumn> *detailColumn*)

Registers a detail column for transferring it when file transfer take place.

### Parameters

- *index*: An integer which controls the order of transferring.

QList<std::shared\_ptr<sh::filesystem::DetailColumn>> **transferrableDetailColumns** ()

A list of all detail columns which are registered becoming transferred in file transfers.

## Private Functions

std::shared\_ptr<sh::filesystem::FilesystemHandler> **\_handler** (std::shared\_ptr<const sh::filesystem::Eurl> *eurl*)

std::shared\_ptr<sh::filesystem::FilesystemOperationProgressMonitor> **\_monitor** (std::shared\_ptr<sh::filesystem::FilesystemHandler> *h*, sh::filesystem::FilesystemOperationProgressMonitor\* *m*)

std::shared\_ptr<const sh::filesystem::Eurl> **\_resolveIfNeeded** (std::shared\_ptr<sh::filesystem::FilesystemHandler> *handler*, std::shared\_ptr<const sh::filesystem::Eurl> *eurl*)

## Private Members

*Operation* \*\_**operation**

## Private Static Attributes

QMap<int, std::shared\_ptr<sh::filesystem::DetailColumn>> **\_detailColumnsMap**

QList<std::shared\_ptr<sh::filesystem::DetailColumn>> **\_detailColumns**

QMutex **\_detailColumnsMutex**

## Friends

**friend class** FilesystemOperationTransfers

**class** MyHandlerTransfer : public sh::filesystem::FilesystemHandler::HandlerTransfer

## Public Functions

**MyHandlerTransfer** (sh::filesystem::FilesystemOperationProgressMonitor \**progressmon* = 0)

**~MyHandlerTransfer** ()

void **respectCancel** ()

void **incrementTransferredBytes** (qint64 *donebytes*)

## Private Members

*sh::filesystem::FilesystemOperationProgressMonitor* \*\_progressmon

**class FilesystemOperationProgressMonitor** : public std::enable\_shared\_from\_this<*FilesystemOperationProgressMonitor*>  
#include <filesystemoperationtransfers.h> Implement this class and use an instance of it as parameter in some methods of *sh::filesystem::FilesystemOperation* for some additional functionality.

This includes monitoring the progress, specifying a resolution when conflicts in the filesystem would occur and more.

It also makes sense to directly instantiate this class in some cases.

Subclassed by *sh::actions::common::ActionAbstractTransferTree::MyFilesystemOperationProgressMonitor*, *sh::scripting::api::ApiFilesystemOperationProgressMonitor*

## Public Functions

**FilesystemOperationProgressMonitor** (*sh::actions::ActionExecutionInfo* \*actionExecution = 0)

Is intended to be directly constructed from everywhere.

### Parameters

- actionExecution: An optional *sh::actions::ActionExecutionInfo* which is used in some places in the default implementation, when available.

bool **hasItemInfo** ()

Checks if this progress monitor provides information about how many items of a certain total number are transferred so far.

quint64 **doneItems** ()

Checks how many items are transferred so far.

quint64 **allItems** ()

Checks how many items are to be transferred in total (as predicted in the current moment).

bool **hasBytesInfo** ()

Checks if this progress monitor provides information about how many byte of a certain total number are transferred so far.

quint64 **doneBytes** ()

Checks how many bytes are transferred so far.

quint64 **allBytes** ()

Checks how many bytes are to be transferred in total (as predicted in the current moment).

QString **getItemInfoFrom** ()

Returns the current source of transfer (as textual information).

QString **getItemInfoTo** ()

Returns the current destination of transfer (as textual information).

QString **estimation** ()

Returns the current performance and time estimation (as textual information).

**~FilesystemOperationProgressMonitor** ()

## Private Functions

void **setProgress** (quint64 *doneitems*, quint64 *allitems*, quint64 *donebytes*, quint64 *allbytes*)  
Used by *FilesystemOperationTransfers* for setting status changes.

void **incProgress** (quint64 *doneitems*, quint64 *allitems*, quint64 *donebytes*, quint64 *allbytes*)  
Used by *FilesystemOperationTransfers* for setting status changes.

void **setItemInfo** (QString *from*, QString *to*)  
Used by *FilesystemOperationTransfers* for setting status changes.

void **setEstimation** (QString *estimation*)  
Used by *FilesystemOperationTransfers* for setting status changes.

void **\_enablestatistics** ()

void **\_computestatistics** ()

void **\_triggerchanged** ()

void **\_stoptriggerchanged** ()

## Private Members

QDateTime **\_laststatistictime**

quint64 **\_laststatisticdonebytes** = 0

quint64 **\_laststatisticdoneitems** = 0

double **\_statisticbytespeed** = 0.0

double **\_statisticitemspeed** = 0.0

int **\_statisticcountdown** = 4

QMutex **\_mutex**

QMutex **\_triggermutex**

quint64 **\_allitems** = 0

quint64 **\_doneitems** = 0

quint64 **\_allbytes** = 0

quint64 **\_donebytes** = 0

bool **\_triggerchanged\_changedrunning** = false

bool **\_triggerchanged\_changedrunagain** = false

bool **\_triggerchanged\_stopped** = false

QString **\_itemfrom**

QString **\_itemto**

QString **\_estimation**

QList<std::shared\_ptr<const *sh::filesystem::Eurl*>> **changes**

bool **\_begancomputestatistics** = false

## Friends

**friend class** FilesystemOperationTransfers

**friend class** FilesystemOperation

**class FilesystemOperationTransfers**

*#include <filesystemoperationtransfers.h>* Used by *FilesystemOperation* for managing the execution of transfer operations.

This class is only used directly by *FilesystemOperation* (but some inner class might be interesting).

## Public Functions

**FilesystemOperationTransfers** (*FilesystemOperation* *\*filesystem*,  
std::shared\_ptr<*FilesystemOperationProgressMonitor*>  
*progressmon*)

Constructed only by the infrastructure and made available otherwise.

void **executestepqueue** ()

Executes the queue.

The execution model is as follows: The complete queue of *OperationStepClass* steps becomes expanded, filesystem conflicts gets resolved and then all steps become executed.

In detail:

At first there is an interleaved process of expanding all steps (see *OperationStepClass::expand*) and checking/resolving filesystem conflicts in the fresh tree (see *doreview*). It checks all steps in the queue for conflicts (e.g. destination already exists). It expands each step which has no conflicts (by enqueueing it to the stepqueue). After those checks, it asks *FilesystemOperationProgressMonitor::resolveConflicts* for a resolution for all outstanding conflicts. With all the steps, which were (or are) in conflict, the entire process becomes repeated. This ends when all steps are expanded and no open conflicts exist anymore.

When this process is done, the execution phase begins. It iterates the queue. It takes the (in FIFO manner) first element (if one is there, otherwise stops). For that step, it makes a late-time conflict check at first. If a conflict appears, it becomes temporarily unexpanded. A resolution loop as above will run for this element, eventually leading to re-expansion. After the check, when no conflicts are open, it executes the step.

void **addMoveItem** (std::shared\_ptr<const *sh::filesystem::Eurl*> *src*, std::shared\_ptr<const *sh::filesystem::Eurl*> *dest*)  
adds one move transfer into the queue.

void **addCopyItem** (std::shared\_ptr<const *sh::filesystem::Eurl*> *src*, std::shared\_ptr<const *sh::filesystem::Eurl*> *dest*)  
adds one copy transfer into the queue.

**~FilesystemOperationTransfers** ()



## Private Functions

```
void _compile_moveItem (std::shared_ptr<const sh::filesystem::Eurl> src,
                        std::shared_ptr<const sh::filesystem::Eurl> dest,
                        QList<OperationStepClass> *result, QList<OperationStepClass>
                        withExpansion)
```

Adds a move transfer step into given list.

```
void _compile_copyItem (std::shared_ptr<const sh::filesystem::Eurl> src,
                        std::shared_ptr<const sh::filesystem::Eurl> dest,
                        QList<OperationStepClass> *result, QList<OperationStepClass>
                        withExpansion)
```

Adds a copy transfer step into given list.

```
void computeProgress ()
```

Computes the current progress (how many items/bytes are transferred from which total?) and notifies the progress monitor.

```
bool doreview (QList<OperationStepClass> reviewsteps)
```

Makes checks if the given queue has conflicts and for asks `resolveConflicts` how to handle them.

For each non-conflicting step, it triggers its expansion, thereby modifying the stepqueue.

It returns when no open conflicts exist.

**Return** If there were any conflicts.

## Private Members

```
QList<OperationStepClass> stepqueue
```

The queue of steps for execution.

```
OperationStepClass *currentstep = 0
```

The step which is currently in execution (or 0).

```
QList<OperationStepClass> donesteps
```

Steps which were already executed.

```
FilesystemOperation *filesystem
```

```
std::shared_ptr<FilesystemOperationProgressMonitor> progressmon
```

```
class OperationStep
```

*#include <filesystemoperationtransfers.h>* One step of execution in a transfer operation by *FilesystemOperation*.

It is typically about a certain source and destination in the filesystem. It also can stay in conflict (due to checks from *FilesystemOperationTransfers::doreview*) and provides methods for resolving them (in *FilesystemOperationTransfers::FilesystemOperationProgressMonitor::resolveConflicts*).

Different subclasses implement distinct kinds of operations (e.g. copying or deleting).

Subclassed by *sh::filesystem::FilesystemOperationTransfers::OperationStepClass*

## Public Types

### **enum ConflictResolution**

Enumeration of ways how to resolve a filesystem conflict.

*Values:*

#### **enumerator Skip**

Skip this element.

#### **enumerator OverwriteDestination**

Overwrite the destination.

#### **enumerator RenameDestinationBefore**

Rename the file at the destination before transferring.

#### **enumerator UseDifferentDestinationName**

Transfer to another destination filename.

#### **enumerator MergeDirectories**

Merge source into destination directory (recursively).

#### **enumerator Unresolved**

No strategy.

#### **enumerator Indirect**

Indirectly solved. Only set by the engine.

## Public Functions

**OperationStep** (*FilesystemOperationTransfers* \*transfers, std::shared\_ptr<const *sh::filesystem::Eurl*> source, std::shared\_ptr<const *sh::filesystem::Eurl*> destination)

Constructed only by the infrastructure and made available otherwise.

QString **conflictDescription** ()

The description of the conflict (if any) as text.

*ConflictResolution* **conflictResolution** ()

The currently chosen conflict resolution.

QString **conflictResolution\_renameDestinationBeforeTo** ()

Returns new destination name (if current conflict resolution is *ConflictResolution::RenameDestinationBefore*).

QString **conflictResolution\_differentDestinationNameTo** ()

Returns new destination name (if current conflict resolution is *ConflictResolution::UseDifferentDestinationName*).

std::shared\_ptr<const *sh::filesystem::Eurl*> **source** ()

Returns the source location to be transferred (if specified).

std::shared\_ptr<const *sh::filesystem::Eurl*> **originalDestination** ()

Returns the destination location (if specified).

This is the complete target path, never the parent directory of the new element.

This is the original destination location. See also effectiveDestination.

std::shared\_ptr<const *sh::filesystem::Eurl*> **effectiveDestination** ()

Returns the real destination location with conflict resolution applied.

```

int sourcetype ()
    Returns the node type of the source.

    Should be overwritten in subclasses whenever it can determine the source type faster than the
    infrastructure.

void setConflictResolve_Skip ()
    Set conflict resolution to ConflictResolution::Skip.

void setConflictResolve_OverwriteDestination ()
    Set conflict resolution to ConflictResolution::OverwriteDestination.

void setConflictResolve_RenameDestinationBefore (QString newname)
    Set conflict resolution to ConflictResolution::RenameDestinationBefore.

void setConflictResolve_UseDifferentDestinationName (QString newname)
    Set conflict resolution to ConflictResolution::UseDifferentDestinationName.

void setConflictResolve_MergeDirectories ()
    Set conflict resolution to ConflictResolution::MergeDirectories.

    See also setConflictResolve_MergeDirectories_isAllowed.

bool setConflictResolve_MergeDirectories_checkAllowed ()
    Checks if conflict resolution ConflictResolution::MergeDirectories is allowed.

~OperationStep ()

```

## Friends

```

friend class FilesystemOperationTransfers

class OperationStep_ApplyDirectoryDetails : public sh::filesystem::FilesystemOperationTransfers::Ope

```

## Public Types

```

enum ConflictResolution
    Enumeration of ways how to resolve a filesystem conflict.

    Values:

    enumerator Skip
        Skip this element.

    enumerator OverwriteDestination
        Overwrite the destination.

    enumerator RenameDestinationBefore
        Rename the file at the destination before transferring.

    enumerator UseDifferentDestinationName
        Transfer to another destination filename.

    enumerator MergeDirectories
        Merge source into destination directory (recursively).

    enumerator Unresolved
        No strategy.

    enumerator Indirect
        Indirectly solved. Only set by the engine.

```

## Public Functions

**OperationStep\_ApplyDirectoryDetails** (*FilesystemOperationTransfers*  
\*transfers, std::shared\_ptr<const  
*sh::filesystem::Eurl*> item,  
QList<QPair<*sh::filesystem::DetailColumn*\*,  
QVariant>> values,  
QList<*OperationStepClass*\*> withEx-  
pansion)

void **execute** (*Operation* \*op)  
Implement this and handle the execution part of this step here.

bool **checkconflicts** (*Operation* \*op)  
Implement this for customizing the conflict checking.

QList<*OperationStepClass*\*> **expand** ()  
Implement this and handle the expansion part of this step here.

bool **hasOwnProgressIncreaseHandling** ()  
If it takes care on its own to signal increases in the transfer progress.

void **\_unexpand** (QList<*OperationStepClass*\*> \*intlist)  
Bookkeeping in *FilesystemOperationTransfers*.

void **\_expand** (QList<*OperationStepClass*\*> \*intlist)  
Bookkeeping in *FilesystemOperationTransfers*.

QList<QPair<*sh::filesystem::DetailColumn*\*, QVariant>> **getFileDetails** (*sh::filesystem::Operation*  
\*op,  
std::shared\_ptr<const  
*sh::filesystem::Eurl*>  
source)

void **applyFileDetails** (*sh::filesystem::Operation* \*op, QList<QPair<*sh::filesystem::DetailColumn*\*,  
QVariant>> values, std::shared\_ptr<const *sh::filesystem::Eurl*>  
destination)

QString **conflictDescription** ()  
The description of the conflict (if any) as text.

*ConflictResolution* **conflictResolution** ()  
The currently chosen conflict resolution.

QString **conflictResolution\_renameDestinationBeforeTo** ()  
Returns new destination name (if current conflict resolution is *ConflictResolu-  
tion::RenameDestinationBefore*).

QString **conflictResolution\_differentDestinationNameTo** ()  
Returns new destination name (if current conflict resolution is *ConflictResolu-  
tion::UseDifferentDestinationName*).

std::shared\_ptr<const *sh::filesystem::Eurl*> **source** ()  
Returns the source location to be transferred (if specified).

std::shared\_ptr<const *sh::filesystem::Eurl*> **originalDestination** ()  
Returns the destination location (if specified).

This is the complete target path, never the parent directory of the new element.

This is the original destination location. See also effectiveDestination.

std::shared\_ptr<const *sh::filesystem::Eurl*> **effectiveDestination** ()

Returns the real destination location with conflict resolution applied.

int **sourcetype** ()

Returns the node type of the source.

Should be overwritten in subclasses whenever it can determine the source type faster than the infrastructure.

void **setConflictResolve\_Skip** ()

Set conflict resolution to *ConflictResolution::Skip*.

void **setConflictResolve\_OverwriteDestination** ()

Set conflict resolution to *ConflictResolution::OverwriteDestination*.

void **setConflictResolve\_RenameDestinationBefore** (QString *newname*)

Set conflict resolution to *ConflictResolution::RenameDestinationBefore*.

void **setConflictResolve\_UseDifferentDestinationName** (QString *newname*)

Set conflict resolution to *ConflictResolution::UseDifferentDestinationName*.

void **setConflictResolve\_MergeDirectories** ()

Set conflict resolution to *ConflictResolution::MergeDirectories*.

See also setConflictResolve\_MergeDirectories\_isAllowed.

bool **setConflictResolve\_MergeDirectories\_checkAllowed** ()

Checks if conflict resolution *ConflictResolution::MergeDirectories* is allowed.

## Public Members

QList<QPair<*sh::filesystem::DetailColumn\**, QVariant>> **\_detailvals**

quint64 **\_cntItems** = 0

Number of items to be transferred in this step (used for progress monitoring).

quint64 **\_cntBytes** = 0

Number of byte to be transferred in this step (used for progress monitoring).

bool **\_isExpanded** = false

Bookkeeping in *FilesystemOperationTransfers*.

QList<*OperationStepClass\**> **\_expandnodes**

Bookkeeping in *FilesystemOperationTransfers*.

*OperationStepClass* \* **\_parentnode** = 0

Bookkeeping in *FilesystemOperationTransfers*.

bool **\_deleteonunexpand** = true

Bookkeeping in *FilesystemOperationTransfers*.

QList<*OperationStepClass\**> **\_withExpansion**

**class OperationStep\_CopyDirectory** : public *sh::filesystem::FilesystemOperationTransfers::OperationStepC*

## Public Types

### **enum ConflictResolution**

Enumeration of ways how to resolve a filesystem conflict.

*Values:*

#### **enumerator Skip**

Skip this element.

#### **enumerator OverwriteDestination**

Overwrite the destination.

#### **enumerator RenameDestinationBefore**

Rename the file at the destination before transferring.

#### **enumerator UseDifferentDestinationName**

Transfer to another destination filename.

#### **enumerator MergeDirectories**

Merge source into destination directory (recursively).

#### **enumerator Unresolved**

No strategy.

#### **enumerator Indirect**

Indirectly solved. Only set by the engine.

## Public Functions

**OperationStep\_CopyDirectory** (*FilesystemOperationTransfers* *\*transfers*,  
std::shared\_ptr<const *sh::filesystem::Eurl*> *src*,  
std::shared\_ptr<const *sh::filesystem::Eurl*> *dest*,  
QList<*OperationStepClass*\*> *withExpansion*)

void **execute** (*Operation* \**op*)

Implement this and handle the execution part of this step here.

QList<*OperationStepClass*\*> **expand** ()

Implement this and handle the expansion part of this step here.

int **sourcetype** ()

Returns the node type of the source.

Should be overwritten in subclasses whenever it can determine the source type faster than the infrastructure.

bool **checkconflicts** (*Operation* \**op*)

Implement this for customizing the conflict checking.

bool **hasOwnProgressIncreaseHandling** ()

If it takes care on its own to signal increases in the transfer progress.

void **\_unexpand** (QList<*OperationStepClass*\*> \**intlist*)

Bookkeeping in *FilesystemOperationTransfers*.

void **\_expand** (QList<*OperationStepClass*\*> \**intlist*)

Bookkeeping in *FilesystemOperationTransfers*.

```

QList<QPair<sh::filesystem::DetailColumn*, QVariant>> getFileDetails (sh::filesystem::Operation
                                                                    *op,
                                                                    std::shared_ptr<const
                                                                    sh::filesystem::Eurl>
                                                                    source)

void applyFileDetails (sh::filesystem::Operation *op, QList<QPair<sh::filesystem::DetailColumn*,
                                                                    QVariant>> values, std::shared_ptr<const sh::filesystem::Eurl>
                                                                    destination)

QString conflictDescription ()
    The description of the conflict (if any) as text.

ConflictResolution conflictResolution ()
    The currently chosen conflict resolution.

QString conflictResolution_renameDestinationBeforeTo ()
    Returns new destination name (if current conflict resolution is ConflictResolution::RenameDestinationBefore).

QString conflictResolution_differentDestinationNameTo ()
    Returns new destination name (if current conflict resolution is ConflictResolution::UseDifferentDestinationName).

std::shared_ptr<const sh::filesystem::Eurl> source ()
    Returns the source location to be transferred (if specified).

std::shared_ptr<const sh::filesystem::Eurl> originalDestination ()
    Returns the destination location (if specified).

    This is the complete target path, never the parent directory of the new element.

    This is the original destination location. See also effectiveDestination.

std::shared_ptr<const sh::filesystem::Eurl> effectiveDestination ()
    Returns the real destination location with conflict resolution applied.

void setConflictResolve_Skip ()
    Set conflict resolution to ConflictResolution::Skip.

void setConflictResolve_OverwriteDestination ()
    Set conflict resolution to ConflictResolution::OverwriteDestination.

void setConflictResolve_RenameDestinationBefore (QString newname)
    Set conflict resolution to ConflictResolution::RenameDestinationBefore.

void setConflictResolve_UseDifferentDestinationName (QString newname)
    Set conflict resolution to ConflictResolution::UseDifferentDestinationName.

void setConflictResolve_MergeDirectories ()
    Set conflict resolution to ConflictResolution::MergeDirectories.

    See also setConflictResolve_MergeDirectories_isAllowed.

bool setConflictResolve_MergeDirectories_checkAllowed ()
    Checks if conflict resolution ConflictResolution::MergeDirectories is allowed.

```

## Public Members

```
quint64 _cntItems = 0
    Number of items to be transferred in this step (used for progress monitoring).

quint64 _cntBytes = 0
    Number of byte to be transferred in this step (used for progress monitoring).

bool _isExpanded = false
    Bookkeeping in FilesystemOperationTransfers.

QList<OperationStepClass*> _expandnodes
    Bookkeeping in FilesystemOperationTransfers.

OperationStepClass *_parentnode = 0
    Bookkeeping in FilesystemOperationTransfers.

bool _deleteonunexpand = true
    Bookkeeping in FilesystemOperationTransfers.

QList<OperationStepClass*> _withExpansion

class OperationStep_CopyFile : public sh::filesystem::FilesystemOperationTransfers::OperationStepClass
```

## Public Types

```
enum ConflictResolution
    Enumeration of ways how to resolve a filesystem conflict.

    Values:

    enumerator Skip
        Skip this element.

    enumerator OverwriteDestination
        Overwrite the destination.

    enumerator RenameDestinationBefore
        Rename the file at the destination before transferring.

    enumerator UseDifferentDestinationName
        Transfer to another destination filename.

    enumerator MergeDirectories
        Merge source into destination directory (recursively).

    enumerator Unresolved
        No strategy.

    enumerator Indirect
        Indirectly solved. Only set by the engine.
```



## Public Functions

**OperationStep\_CopyFile** (*FilesystemOperationTransfers* *\*transfers*,  
 std::shared\_ptr<const *sh::filesystem::Eurl*> *src*,  
 std::shared\_ptr<const *sh::filesystem::Eurl*> *dest*,  
 QList<*OperationStepClass*\*> *withExpansion*)

void **execute** (*Operation* \**op*)  
 Implement this and handle the execution part of this step here.

int **sourcetype** ()  
 Returns the node type of the source.  
 Should be overwritten in subclasses whenever it can determine the source type faster than the infrastructure.

bool **hasOwnProgressIncreaseHandling** ()  
 If it takes care on its own to signal increases in the transfer progress.

QList<*OperationStepClass*\*> **expand** ()  
 Implement this and handle the expansion part of this step here.

bool **checkconflicts** (*Operation* \**op*)  
 Implement this for customizing the conflict checking.

void **\_unexpand** (QList<*OperationStepClass*\*> \**intlist*)  
 Bookkeeping in *FilesystemOperationTransfers*.

void **\_expand** (QList<*OperationStepClass*\*> \**intlist*)  
 Bookkeeping in *FilesystemOperationTransfers*.

QList<QPair<*sh::filesystem::DetailColumn*\*, QVariant>> **getFileDetails** (*sh::filesystem::Operation* \**op*,  
 std::shared\_ptr<const *sh::filesystem::Eurl*> *source*)

void **applyFileDetails** (*sh::filesystem::Operation* \**op*, QList<QPair<*sh::filesystem::DetailColumn*\*,  
 QVariant>> *values*, std::shared\_ptr<const *sh::filesystem::Eurl*> *destination*)

QString **conflictDescription** ()  
 The description of the conflict (if any) as text.

*ConflictResolution* **conflictResolution** ()  
 The currently chosen conflict resolution.

QString **conflictResolution\_renameDestinationBeforeTo** ()  
 Returns new destination name (if current conflict resolution is *ConflictResolution::RenameDestinationBefore*).

QString **conflictResolution\_differentDestinationNameTo** ()  
 Returns new destination name (if current conflict resolution is *ConflictResolution::UseDifferentDestinationName*).

std::shared\_ptr<const *sh::filesystem::Eurl*> **source** ()  
 Returns the source location to be transferred (if specified).

std::shared\_ptr<const *sh::filesystem::Eurl*> **originalDestination** ()  
 Returns the destination location (if specified).  
 This is the complete target path, never the parent directory of the new element.  
 This is the original destination location. See also *effectiveDestination*.

`std::shared_ptr<const sh::filesystem::Eurl> effectiveDestination ()`  
Returns the real destination location with conflict resolution applied.

`void setConflictResolve_Skip ()`  
Set conflict resolution to *ConflictResolution::Skip*.

`void setConflictResolve_OverwriteDestination ()`  
Set conflict resolution to *ConflictResolution::OverwriteDestination*.

`void setConflictResolve_RenameDestinationBefore (QString newname)`  
Set conflict resolution to *ConflictResolution::RenameDestinationBefore*.

`void setConflictResolve_UseDifferentDestinationName (QString newname)`  
Set conflict resolution to *ConflictResolution::UseDifferentDestinationName*.

`void setConflictResolve_MergeDirectories ()`  
Set conflict resolution to *ConflictResolution::MergeDirectories*.

See also `setConflictResolve_MergeDirectories_isAllowed`.

`bool setConflictResolve_MergeDirectories_checkAllowed ()`  
Checks if conflict resolution *ConflictResolution::MergeDirectories* is allowed.

## Public Members

`quint64 _cntItems = 0`  
Number of items to be transferred in this step (used for progress monitoring).

`quint64 _cntBytes = 0`  
Number of byte to be transferred in this step (used for progress monitoring).

`bool _isExpanded = false`  
Bookkeeping in *FilesystemOperationTransfers*.

`QList<OperationStepClass*> _expandnodes`  
Bookkeeping in *FilesystemOperationTransfers*.

`OperationStepClass *_parentnode = 0`  
Bookkeeping in *FilesystemOperationTransfers*.

`bool _deleteonunexpand = true`  
Bookkeeping in *FilesystemOperationTransfers*.

`QList<OperationStepClass*> _withExpansion`

`class OperationStep_CopyLink : public sh::filesystem::FilesystemOperationTransfers::OperationStepClass`

## Public Types

`enum ConflictResolution`  
Enumeration of ways how to resolve a filesystem conflict.

Values:

`enumerator Skip`  
Skip this element.

`enumerator OverwriteDestination`  
Overwrite the destination.

`enumerator RenameDestinationBefore`  
Rename the file at the destination before transferring.

**enumerator UseDifferentDestinationName**

Transfer to another destination filename.

**enumerator MergeDirectories**

Merge source into destination directory (recursively).

**enumerator Unresolved**

No strategy.

**enumerator Indirect**

Indirectly solved. Only set by the engine.

## Public Functions

**OperationStep\_CopyLink** (*FilesystemOperationTransfers* *\*transfers*,  
std::shared\_ptr<const *sh::filesystem::Eurl*> *src*,  
std::shared\_ptr<const *sh::filesystem::Eurl*> *dest*,  
QList<*OperationStepClass*\*> *withExpansion*)

void **execute** (*Operation* \**op*)

Implement this and handle the execution part of this step here.

int **sourcetype** ()

Returns the node type of the source.

Should be overwritten in subclasses whenever it can determine the source type faster than the infrastructure.

QList<*OperationStepClass*\*> **expand** ()

Implement this and handle the expansion part of this step here.

bool **checkconflicts** (*Operation* \**op*)

Implement this for customizing the conflict checking.

bool **hasOwnProgressIncreaseHandling** ()

If it takes care on its own to signal increases in the transfer progress.

void **\_unexpand** (QList<*OperationStepClass*\*> \**intlist*)

Bookkeeping in *FilesystemOperationTransfers*.

void **\_expand** (QList<*OperationStepClass*\*> \**intlist*)

Bookkeeping in *FilesystemOperationTransfers*.

QList<QPair<*sh::filesystem::DetailColumn*\*, QVariant>> **getFileDetails** (*sh::filesystem::Operation* \**op*,  
std::shared\_ptr<const *sh::filesystem::Eurl*>  
*source*)

void **applyFileDetails** (*sh::filesystem::Operation* \**op*, QList<QPair<*sh::filesystem::DetailColumn*\*,  
QVariant>> *values*, std::shared\_ptr<const *sh::filesystem::Eurl*>  
*destination*)

QString **conflictDescription** ()

The description of the conflict (if any) as text.

*ConflictResolution* **conflictResolution** ()

The currently chosen conflict resolution.

QString **conflictResolution\_renameDestinationBeforeTo** ()

Returns new destination name (if current conflict resolution is *ConflictResolution::RenameDestinationBefore*).

QString **conflictResolution\_differentDestinationNameTo** ()  
Returns new destination name (if current conflict resolution is *ConflictResolution::UseDifferentDestinationName*).

std::shared\_ptr<const *sh::filesystem::Eurl*> **source** ()  
Returns the source location to be transferred (if specified).

std::shared\_ptr<const *sh::filesystem::Eurl*> **originalDestination** ()  
Returns the destination location (if specified).  
  
This is the complete target path, never the parent directory of the new element.  
  
This is the original destination location. See also *effectiveDestination*.

std::shared\_ptr<const *sh::filesystem::Eurl*> **effectiveDestination** ()  
Returns the real destination location with conflict resolution applied.

void **setConflictResolve\_Skip** ()  
Set conflict resolution to *ConflictResolution::Skip*.

void **setConflictResolve\_OverwriteDestination** ()  
Set conflict resolution to *ConflictResolution::OverwriteDestination*.

void **setConflictResolve\_RenameDestinationBefore** (QString *newname*)  
Set conflict resolution to *ConflictResolution::RenameDestinationBefore*.

void **setConflictResolve\_UseDifferentDestinationName** (QString *newname*)  
Set conflict resolution to *ConflictResolution::UseDifferentDestinationName*.

void **setConflictResolve\_MergeDirectories** ()  
Set conflict resolution to *ConflictResolution::MergeDirectories*.  
  
See also *setConflictResolve\_MergeDirectories\_isAllowed*.

bool **setConflictResolve\_MergeDirectories\_checkAllowed** ()  
Checks if conflict resolution *ConflictResolution::MergeDirectories* is allowed.

## Public Members

quint64 **\_cntItems** = 0  
Number of items to be transferred in this step (used for progress monitoring).

quint64 **\_cntBytes** = 0  
Number of byte to be transferred in this step (used for progress monitoring).

bool **\_isExpanded** = false  
Bookkeeping in *FilesystemOperationTransfers*.

QList<*OperationStepClass*\*> **\_expandnodes**  
Bookkeeping in *FilesystemOperationTransfers*.

*OperationStepClass* \* **\_parentnode** = 0  
Bookkeeping in *FilesystemOperationTransfers*.

bool **\_deleteonunexpand** = true  
Bookkeeping in *FilesystemOperationTransfers*.

QList<*OperationStepClass*\*> **\_withExpansion**

**class OperationStep\_DeleteItem: public *sh::filesystem::FilesystemOperationTransfers::OperationStepClass***

## Public Types

### enum ConflictResolution

Enumeration of ways how to resolve a filesystem conflict.

*Values:*

#### enumerator Skip

Skip this element.

#### enumerator OverwriteDestination

Overwrite the destination.

#### enumerator RenameDestinationBefore

Rename the file at the destination before transferring.

#### enumerator UseDifferentDestinationName

Transfer to another destination filename.

#### enumerator MergeDirectories

Merge source into destination directory (recursively).

#### enumerator Unresolved

No strategy.

#### enumerator Indirect

Indirectly solved. Only set by the engine.

## Public Functions

```
OperationStep_DeleteItem (FilesystemOperationTransfers *transfers,
                          std::shared_ptr<const sh::filesystem::Eurl> src,
                          std::shared_ptr<const sh::filesystem::Eurl> dest,
                          QList<OperationStepClass*> withExpansion)
```

```
void execute (Operation *op)
```

Implement this and handle the execution part of this step here.

```
bool checkconflicts (Operation *op)
```

Implement this for customizing the conflict checking.

```
int sourcetype ()
```

Returns the node type of the source.

Should be overwritten in subclasses whenever it can determine the source type faster than the infrastructure.

```
QList<OperationStepClass*> expand ()
```

Implement this and handle the expansion part of this step here.

```
bool hasOwnProgressIncreaseHandling ()
```

If it takes care on its own to signal increases in the transfer progress.

```
void _unexpand (QList<OperationStepClass*> *intlist)
```

Bookkeeping in *FilesystemOperationTransfers*.

```
void _expand (QList<OperationStepClass*> *intlist)
```

Bookkeeping in *FilesystemOperationTransfers*.

`QList<QPair<sh::filesystem::DetailColumn*, QVariant>> getFileDetails (sh::filesystem::Operation`  
`*op,`  
`std::shared_ptr<const`  
`sh::filesystem::Eurl>`  
`source)`

`void applyFileDetails (sh::filesystem::Operation *op, QList<QPair<sh::filesystem::DetailColumn*,`  
`QVariant>> values, std::shared_ptr<const sh::filesystem::Eurl>`  
`destination)`

`QString conflictDescription ()`  
The description of the conflict (if any) as text.

`ConflictResolution conflictResolution ()`  
The currently chosen conflict resolution.

`QString conflictResolution_renameDestinationBeforeTo ()`  
Returns new destination name (if current conflict resolution is *ConflictResolution::RenameDestinationBefore*).

`QString conflictResolution_differentDestinationNameTo ()`  
Returns new destination name (if current conflict resolution is *ConflictResolution::UseDifferentDestinationName*).

`std::shared_ptr<const sh::filesystem::Eurl> source ()`  
Returns the source location to be transferred (if specified).

`std::shared_ptr<const sh::filesystem::Eurl> originalDestination ()`  
Returns the destination location (if specified).

This is the complete target path, never the parent directory of the new element.

This is the original destination location. See also `effectiveDestination`.

`std::shared_ptr<const sh::filesystem::Eurl> effectiveDestination ()`  
Returns the real destination location with conflict resolution applied.

`void setConflictResolve_Skip ()`  
Set conflict resolution to *ConflictResolution::Skip*.

`void setConflictResolve_OverwriteDestination ()`  
Set conflict resolution to *ConflictResolution::OverwriteDestination*.

`void setConflictResolve_RenameDestinationBefore (QString newname)`  
Set conflict resolution to *ConflictResolution::RenameDestinationBefore*.

`void setConflictResolve_UseDifferentDestinationName (QString newname)`  
Set conflict resolution to *ConflictResolution::UseDifferentDestinationName*.

`void setConflictResolve_MergeDirectories ()`  
Set conflict resolution to *ConflictResolution::MergeDirectories*.

See also `setConflictResolve_MergeDirectories_isAllowed`.

`bool setConflictResolve_MergeDirectories_checkAllowed ()`  
Checks if conflict resolution *ConflictResolution::MergeDirectories* is allowed.

## Public Members

```

quint64 _cntItems = 0
    Number of items to be transferred in this step (used for progress monitoring).

quint64 _cntBytes = 0
    Number of byte to be transferred in this step (used for progress monitoring).

bool _isExpanded = false
    Bookkeeping in FilesystemOperationTransfers.

QList<OperationStepClass*> _expandnodes
    Bookkeeping in FilesystemOperationTransfers.

OperationStepClass *_parentnode = 0
    Bookkeeping in FilesystemOperationTransfers.

bool _deleteonunexpand = true
    Bookkeeping in FilesystemOperationTransfers.

QList<OperationStepClass*> _withExpansion

class OperationStep_RenameItem: public sh::filesystem::FilesystemOperationTransfers::OperationStepClass

```

## Public Types

```

enum ConflictResolution
    Enumeration of ways how to resolve a filesystem conflict.

    Values:

    enumerator Skip
        Skip this element.

    enumerator OverwriteDestination
        Overwrite the destination.

    enumerator RenameDestinationBefore
        Rename the file at the destination before transferring.

    enumerator UseDifferentDestinationName
        Transfer to another destination filename.

    enumerator MergeDirectories
        Merge source into destination directory (recursively).

    enumerator Unresolved
        No strategy.

    enumerator Indirect
        Indirectly solved. Only set by the engine.

```

## Public Functions

**OperationStep\_RenameItem** (*FilesystemOperationTransfers* *\*transfers*,  
std::shared\_ptr<const *sh::filesystem::Eurl*> *src*, QString  
*dstpath*, QList<*OperationStepClass*> *withExpansion*)

QList<*OperationStepClass*> **expand** ()  
Implement this and handle the expansion part of this step here.

void **execute** (*Operation* *\*op*)  
Implement this and handle the execution part of this step here.

bool **checkconflicts** (*Operation* *\*op*)  
Implement this for customizing the conflict checking.

bool **hasOwnProgressIncreaseHandling** ()  
If it takes care on its own to signal increases in the transfer progress.

void **\_unexpand** (QList<*OperationStepClass*> *\*intlist*)  
Bookkeeping in *FilesystemOperationTransfers*.

void **\_expand** (QList<*OperationStepClass*> *\*intlist*)  
Bookkeeping in *FilesystemOperationTransfers*.

QList<QPair<*sh::filesystem::DetailColumn*\*, QVariant>> **getFileDetails** (*sh::filesystem::Operation*  
*\*op*,  
std::shared\_ptr<const  
*sh::filesystem::Eurl*>  
*source*)

void **applyFileDetails** (*sh::filesystem::Operation* *\*op*, QList<QPair<*sh::filesystem::DetailColumn*\*,  
QVariant>> *values*, std::shared\_ptr<const *sh::filesystem::Eurl*>  
*destination*)

QString **conflictDescription** ()  
The description of the conflict (if any) as text.

*ConflictResolution* **conflictResolution** ()  
The currently chosen conflict resolution.

QString **conflictResolution\_renameDestinationBeforeTo** ()  
Returns new destination name (if current conflict resolution is *ConflictResolution::RenameDestinationBefore*).

QString **conflictResolution\_differentDestinationNameTo** ()  
Returns new destination name (if current conflict resolution is *ConflictResolution::UseDifferentDestinationName*).

std::shared\_ptr<const *sh::filesystem::Eurl*> **source** ()  
Returns the source location to be transferred (if specified).

std::shared\_ptr<const *sh::filesystem::Eurl*> **originalDestination** ()  
Returns the destination location (if specified).  
  
This is the complete target path, never the parent directory of the new element.  
  
This is the original destination location. See also *effectiveDestination*.

std::shared\_ptr<const *sh::filesystem::Eurl*> **effectiveDestination** ()  
Returns the real destination location with conflict resolution applied.

int **sourcetype** ()  
Returns the node type of the source.



Should be overwritten in subclasses whenever it can determine the source type faster than the infrastructure.

```
void setConflictResolve_Skip ()
    Set conflict resolution to ConflictResolution::Skip.

void setConflictResolve_OverwriteDestination ()
    Set conflict resolution to ConflictResolution::OverwriteDestination.

void setConflictResolve_RenameDestinationBefore (QString newname)
    Set conflict resolution to ConflictResolution::RenameDestinationBefore.

void setConflictResolve_UseDifferentDestinationName (QString newname)
    Set conflict resolution to ConflictResolution::UseDifferentDestinationName.

void setConflictResolve_MergeDirectories ()
    Set conflict resolution to ConflictResolution::MergeDirectories.

    See also setConflictResolve_MergeDirectories_isAllowed.

bool setConflictResolve_MergeDirectories_checkAllowed ()
    Checks if conflict resolution ConflictResolution::MergeDirectories is allowed.
```

## Public Members

```
quint64 _cntItems = 0
    Number of items to be transferred in this step (used for progress monitoring).

quint64 _cntBytes = 0
    Number of byte to be transferred in this step (used for progress monitoring).

bool _isExpanded = false
    Bookkeeping in FilesystemOperationTransfers.

QList<OperationStepClass*> _expandnodes
    Bookkeeping in FilesystemOperationTransfers.

OperationStepClass * _parentnode = 0
    Bookkeeping in FilesystemOperationTransfers.

bool _deleteonunexpand = true
    Bookkeeping in FilesystemOperationTransfers.

QList<OperationStepClass*> _withExpansion
```

```
class OperationStepClass : public sh::filesystem::FilesystemOperationTransfers::OperationStep
    A base class for all implementations.
```

They are used from inside *FilesystemOperationTransfers::executestepqueue*.

Subclassed by *sh::filesystem::FilesystemOperationTransfers::OperationStep\_ApplyDirectoryDetails*,  
*sh::filesystem::FilesystemOperationTransfers::OperationStep\_CopyDirectory*,  
*sh::filesystem::FilesystemOperationTransfers::OperationStep\_CopyFile*,  
*sh::filesystem::FilesystemOperationTransfers::OperationStep\_CopyLink*,  
*sh::filesystem::FilesystemOperationTransfers::OperationStep\_DeleteItem*,  
*sh::filesystem::FilesystemOperationTransfers::OperationStep\_RenameItem*

## Public Types

### **enum ConflictResolution**

Enumeration of ways how to resolve a filesystem conflict.

*Values:*

#### **enumerator Skip**

Skip this element.

#### **enumerator OverwriteDestination**

Overwrite the destination.

#### **enumerator RenameDestinationBefore**

Rename the file at the destination before transferring.

#### **enumerator UseDifferentDestinationName**

Transfer to another destination filename.

#### **enumerator MergeDirectories**

Merge source into destination directory (recursively).

#### **enumerator Unresolved**

No strategy.

#### **enumerator Indirect**

Indirectly solved. Only set by the engine.

## Public Functions

**OperationStepClass** (*FilesystemOperationTransfers* \*transfers, std::shared\_ptr<const *sh::filesystem::Eurl*> source, std::shared\_ptr<const *Eurl*> destination, QList<*OperationStepClass*\*> withExpansion)

void **execute** (*Operation* \*op) = 0

Implement this and handle the execution part of this step here.

QList<*OperationStepClass*\*> **expand** ()

Implement this and handle the expansion part of this step here.

bool **checkconflicts** (*Operation* \*op)

Implement this for customizing the conflict checking.

bool **hasOwnProgressIncreaseHandling** ()

If it takes care on its own to signal increases in the transfer progress.

void **\_unexpand** (QList<*OperationStepClass*\*> \*intlist)

Bookkeeping in *FilesystemOperationTransfers*.

void **\_expand** (QList<*OperationStepClass*\*> \*intlist)

Bookkeeping in *FilesystemOperationTransfers*.

**~OperationStepClass** ()

QList<QPair<*sh::filesystem::DetailColumn*\*, QVariant>> **getFileDetails** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> source)

```

void applyFileDetails (sh::filesystem::Operation *op, QList<QPair<sh::filesystem::DetailColumn*,
    QVariant>> values, std::shared_ptr<const sh::filesystem::Eurl>
    destination)

QString conflictDescription ()
    The description of the conflict (if any) as text.

ConflictResolution conflictResolution ()
    The currently chosen conflict resolution.

QString conflictResolution_renameDestinationBeforeTo ()
    Returns new destination name (if current conflict resolution is ConflictResolution::RenameDestinationBefore).

QString conflictResolution_differentDestinationNameTo ()
    Returns new destination name (if current conflict resolution is ConflictResolution::UseDifferentDestinationName).

std::shared_ptr<const sh::filesystem::Eurl> source ()
    Returns the source location to be transferred (if specified).

std::shared_ptr<const sh::filesystem::Eurl> originalDestination ()
    Returns the destination location (if specified).

    This is the complete target path, never the parent directory of the new element.

    This is the original destination location. See also effectiveDestination.

std::shared_ptr<const sh::filesystem::Eurl> effectiveDestination ()
    Returns the real destination location with conflict resolution applied.

int sourcetype ()
    Returns the node type of the source.

    Should be overwritten in subclasses whenever it can determine the source type faster than the
    infrastructure.

void setConflictResolve_Skip ()
    Set conflict resolution to ConflictResolution::Skip.

void setConflictResolve_OverwriteDestination ()
    Set conflict resolution to ConflictResolution::OverwriteDestination.

void setConflictResolve_RenameDestinationBefore (QString newname)
    Set conflict resolution to ConflictResolution::RenameDestinationBefore.

void setConflictResolve_UseDifferentDestinationName (QString newname)
    Set conflict resolution to ConflictResolution::UseDifferentDestinationName.

void setConflictResolve_MergeDirectories ()
    Set conflict resolution to ConflictResolution::MergeDirectories.

    See also setConflictResolve_MergeDirectories_isAllowed.

bool setConflictResolve_MergeDirectories_checkAllowed ()
    Checks if conflict resolution ConflictResolution::MergeDirectories is allowed.

```

### Public Members

quint64 **\_cntItems** = 0  
Number of items to be transferred in this step (used for progress monitoring).

quint64 **\_cntBytes** = 0  
Number of byte to be transferred in this step (used for progress monitoring).

bool **\_isExpanded** = false  
Bookkeeping in *FilesystemOperationTransfers*.

QList<*OperationStepClass*\*> **\_expandnodes**  
Bookkeeping in *FilesystemOperationTransfers*.

*OperationStepClass* \***\_parentnode** = 0  
Bookkeeping in *FilesystemOperationTransfers*.

bool **\_deleteonunexpand** = true  
Bookkeeping in *FilesystemOperationTransfers*.

QList<*OperationStepClass*\*> **\_withExpansion**

**class SingleStepMonitor : public *sh::filesystem::FilesystemHandler::HandlerTransfer***  
Used for managing detailed progress changes about a single step.

### Public Functions

**SingleStepMonitor** (*OperationStepClass* \*step)

void **respectCancel** ()

void **incrementTransferredBytes** (quint64 donebytes)

**~SingleStepMonitor** ()

### Private Members

*OperationStepClass* \***step**

quint64 **\_donebytes** = 0

**class LoadOnDemandPlaceholderFilesystemNode : public *sh::filesystem::FilesystemNode***  
*#include <filesystemnode.h>* A special node, which is unselectable and shows a 'loading' label.

It does not correspond to a file, directory or similar which actually exists. It is also never parent or a child of another node.

They are used internally by the filesystem model.

## Public Functions

QString **displayName** ()

Returns the display name (what the user interface displays).

void **requestDetails** (bool *force*)

Requests to fetch details for this node.

**LoadOnDemandPlaceholderFilesystemNode** (std::shared\_ptr<sh::filesystem::FilesystemNode>  
parent)

Constructed only by the infrastructure and made available otherwise.

std::shared\_ptr<const sh::filesystem::Eurl> **eurl** ()

Returns the sh::filesystem::Eurl entry address.

sh::filesystem::FilesystemModel \***model** ()

Convenience shortcut to the sh::filesystem::FilesystemModel.

void **setDisplayname** (QString *displayname*)

Sets the display name (what the user interface displays).

sh::filesystem::FilesystemHandler \***handler** ()

Returns the handler which is responsible for this node. This should be the same as  
sh::filesystem::FilesystemHandlerRegister.findHandler(eurl.scheme).

int **nodetype** ()

Returns the FilesystemNodeType node type.

int **childnodeCount** ()

Returns the number of children nodes.

std::shared\_ptr<sh::filesystem::FilesystemNode> **childnode** (int *i*)

Returns i-th child node.

sh::filesystem::ModelBackedFilesystemNodeList \***childnodes** ()

Returns the list of children.

std::shared\_ptr<sh::filesystem::FilesystemNode> **parentnode** () const

Returns the parent node.

This is 0 if the node does not exist in the model (i.e. not yet inserted or removed afterwards). In most cases, this can be interpreted as ‘file currently does not exist’.

int **index** ()

Returns the index of this node in the parent’s list of children.

void **addChild** (std::shared\_ptr<sh::filesystem::FilesystemNode> *node*)

Adds a new child. Not allowed to call more than once for a node.

void **removeChild** (std::shared\_ptr<sh::filesystem::FilesystemNode> *node*)

Removes a new child. Not allowed to call more than once for a node.

bool **isFetchingSubitems** ()

Checks if currently subitems are fetched (i.e. the ‘loading’ node is shown).

QIcon **icon** ()

Returns the node icon.

void **setIcon** (QIcon *icon*)

Sets the node icon.

bool **isHidden** ()

Returns if the node is hidden.

Note: It's not difficult for the user to also show the hidden items.

void **setHidden** (bool *v*)  
Sets if the node is hidden.

See also *isHidden()*.

void **addDetail** (std::shared\_ptr<*sh::filesystem::DetailColumn*> *column*)  
Adds a detail to this node.

std::shared\_ptr<*sh::filesystem::DetailColumn*> **getDetailColumnByIndex** (int *index*)  
Returns the *i*-th detail column registered to this node.

int **getDetailColumnsCount** () **const**  
Returns the number of detail columns registered to this node.

bool **isRootNode** () **const**  
Checks if this is the root node of the model.

bool **isConfigured** () **const**  
Checks if this node was already configured by a handler.

bool **isAlive** () **const**  
Checks if this node currently exists in the model.

QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> **linkDestinationNodes** () **const**  
Returns the list of link source nodes. If this node is a link, the link destination nodes may influence the behavior and appearance of this node.

This information does not come from inside but must be set from outside the model implementation.

void **setLinkDestinationNodes** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>>  
*linkdestinations*)  
Sets the list of link destination nodes. If this node is a link, the link destination nodes may influence the behavior and appearance of this node.

void **requestContainedItems** (*sh::filesystem::FilesystemNodeType* *type* = *FilesystemNode-*  
*Type::NONE*)  
Requests to fetch the children of this node with help of the filesystem handler.

## Signals

void **removed** ()  
Is emitted when this node is not placed in the tree anymore. This does not mean the physical deletion of the instance, but just means it is not alive from now on.

void **\_detailsAvailable** ()  
Is emitted when values for detail columns arrived.

void **iconChanged** ()  
Is emitted when the icon changed.

void **isHiddenChanged** ()  
Is emitted when the hidden flag changed.

## Private Members

bool **loading**

## Friends

**friend class** *sh::filesystem::FilesystemModel*

**class** **ModelBackedFilesystemNodeList** : public *sh::filesystem::FilesystemNodeList*  
*#include <filesystemodelist.h>* This *FilesystemNodeList* subclass builds a part of the filesystem model.

The listed nodes are actually owned and handled by a *sh::filesystem::FilesystemNode*. Node additions and removals will directly influence the model.

## Public Functions

**ModelBackedFilesystemNodeList** (*sh::filesystem::FilesystemNode \*node*)

Constructed only internally.

void **addItem** (QSet<std::shared\_ptr<*FilesystemNode*>> *nodes*)

void **addItem** (std::shared\_ptr<*FilesystemNode*> *node*)

void **removeItems** (QSet<std::shared\_ptr<*FilesystemNode*>> *nodes*)

void **removeItem** (std::shared\_ptr<*FilesystemNode*> *node*)

void **resetItems** (*sh::filesystem::FilesystemNodeType* *type*, QSet<std::shared\_ptr<*FilesystemNode*>> *nodes*)

std::shared\_ptr<*FilesystemNode*> **myNode** ()

Returns the node which owns this children list. The result may be 0 for non model-backed lists.

void **clear** ()

Clears the entire list. This is not a high-level operation, but something only used internally (not part of the interface).

void **addPermanentItem** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> *node*)

Adds an item to this list. In a model-backed list, this triggers the internal model mounting calls. It is not allowed to call this method twice with the same node.

This list is safe against removing from later item lists. Only **removePermanentItem** removes it. See also **addItem**.

void **removePermanentItem** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> *node*)

Removes a permanent node from this list. In a model-backed list, this triggers the internal model unmounting calls. It is not allowed to call this method with a node, which is not contained in that list.

void **addItem** (QSet<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> *nodes*) = 0

Adds items to this list. In a model-backed list, this triggers the internal model mounting calls. It is not allowed to call this method twice with the same node.

void **addItem** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> *node*) = 0

Adds an item to this list. In a model-backed list, this triggers the internal model mounting calls. It is not allowed to call this method twice with the same node.

void **removeItems** (QSet<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> *nodes*) = 0

Removes nodes from this list. In a model-backed list, this triggers the internal model unmounting calls. It is not allowed to call this method with a node, which is not contained in that list.

void **removeItem** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> node) = 0  
Removes a node from this list. In a model-backed list, this triggers the internal model unmounting calls. It is not allowed to call this method with a node, which is not contained in that list.

void **resetItems** (*sh::filesystem::FilesystemNodeType* type, QSet<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> nodes) = 0  
Resets the list for a given node type to a given new list of children nodes.

bool **contains** (std::shared\_ptr<*FilesystemNode*> node)  
Checks if this list contains a given node.

const QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> \***nodes** ()  
Returns the list of nodes currently stored in this instance.

### Private Functions

void **addItem\_helper** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> node)

### Private Members

QSet<*sh::filesystem::FilesystemNode*\*> **\_permanentnodes**

*sh::filesystem::FilesystemNode* \***\_mynode**

*sh::filesystem::FilesystemModel* \***mymodel**

**class Operation : public QObject**

*#include <operation.h>* A operation surrounds a completed series of steps in the filesystem.

For some steps to execute in the filesystem (i.e. not only the local one but the entire *Eurl* universe), it fetches intermediate files, caches them and writes them back to their real location afterwards.

It provides transaction-like operations for committing/dropping those intermediate files. A high-level interface for file management is available in *filesystem*.

It is allowed to create new instances from scratch, but you should check if you implicitly have an instance available in your situation, or if *sh::tools::OperationsCache* is an option.

### Public Functions

**Operation** (QObject \*parent = 0)

Often constructed by the infrastructure and made available, but can also be constructed directly.

quint64 **getFreeCacheSpace** ()

Returns the free disk space (in bytes) available for operations like *fetchContainerFile()* or *fetchFile()*.

QString **fetchContainerFile** (std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QString namehint)

Fetches the container file for a eurl locally and returns the local path. Use this with care and only if needed. In many cases working with streams is the better way!

QString **fetchContainerFile** (std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)

QString **fetchFile** (std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QString namehint)

Fetches a file locally and returns the local path. Use this with care and only if needed. In many cases working with streams is the better way!

QString **fetchFile** (std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)



```

void abort ()
    Aborts transaction dropping all pending changes.

void commit ()
    Commits transaction applying all pending changes.

void enableDetailsCache ()
    Enable details caching.

bool isDetailsCacheEnabled ()
    Is details caching enabled?

void storeDetailInCache (std::shared_ptr<const sh::filesystem::Eurl> eurl,
                        sh::filesystem::DetailColumn *column, QVariant value)
    Stores a column detail in cache.

QVariant getDetailFromCache (std::shared_ptr<const sh::filesystem::Eurl> eurl, const
                            sh::filesystem::DetailColumn *column)
    Gets a column value from cache.

QList<std::shared_ptr<const sh::filesystem::Eurl>> pendingCommits ()
    Returns a list of sh::filesystem::Eurl items which are marked for commit.

QString getTempDir ()
    Creates a fresh temporary folder and returns the path to it.

void setCustomData (std::shared_ptr<const sh::filesystem::Eurl> eurl, QString key, QVariant
                    data)
    Stores custom data.

QVariant getCustomData (std::shared_ptr<const sh::filesystem::Eurl> eurl, QString key)
    Gets stored custom data.

void setMaxAllowedSizeRatioPerPart (double v)
    Sets a maximum part of available disk space for usage. This is exotic functionality you typically don't
    need. .

Parameters
    • v: A ratio (between 0 and 1) of the available disk space which one part maximally may cost.

sh::filesystem::FilesystemOperation *filesystem ()
    Gets the sh::filesystem::FilesystemOperation filesystem operation object.

~Operation ()

```

## Public Static Functions

```

void writeIODeviceToFile (QIODevice *content, QString filepath)
    Low-level function for writing a QIODevice content to a local path.

```

### Private Members

```
QHash<std::shared_ptr<const sh::filesystem::Eurl>, QString> files
QSet<std::shared_ptr<const sh::filesystem::Eurl>> fileIsTemporary
QList<QString> tempDirs
bool _detailsCacheEnabled = false
QHash<std::shared_ptr<const sh::filesystem::Eurl>, QMap<QString, QVariant>> _customData
QHash<const sh::filesystem::DetailColumn*, QHash<std::shared_ptr<const sh::filesystem::Eurl>, QVariant>> _detail
sh::filesystem::FilesystemOperation *_filesystemOperation
double _maxAllowedSizeRatioPerPart
QMutex operationlock
```

### Private Static Attributes

```
QMutex _tempfilemutex
QString _tempdir
class MaxAllowedSizeRatioPerPartExceededException : public sh::exceptions::IOException
    #include <operation.h> IO exception raised when the value in setMaxAllowedSizeRatioPerPart() was
    exceeded.
```

### Public Functions

```
MaxAllowedSizeRatioPerPartExceededException()
QString message() const
QString name() const
QString classes() const
QString details() const
QString callstack() const
QString auxiliary() const
QString customValue(QString key) const
bool isRuntimeException() const
bool isProgramException() const
bool isRetryable() const
bool isResumeable() const
bool isDetailsAreInteresting() const
int autoRetryRecommendedIn() const
void setResumeable(bool v)
void setReTryable(bool v)
void setCustomValue(QString key, QString value)
```

```
bool isClass (QString classname)
sh::exceptions::ExceptionData data ()
```

### Public Static Functions

```
template<class Handler>
std::shared_ptr<RegisterHandler<Handler>> createRegisterHandler (Handler    han-
                                                                dlr,        QS-
                                                                tack<Handler>
                                                                *stack)
```

```
void executeGuarded (std::function<void>
    > fct int flags = 0 Executes some code with some standard exection handling around.
```

#### Parameters

- *fct*: The code to execute guarded.
- *flags*: Flags of *ExecuteGuardFlag* for choosing the behavior.

```
void executeGuarded_errorpanel (std::function<void>
    > fct int flags = 0
```

```
QString _demangle (QString l)
```

```
void exceptionDialog (sh::exceptions::Exception &e, bool *doRetry)
```

### Public Static Attributes

```
const QString UNDEFINED_ERROR_OCCURRED = QObject::tr("An unspecified error occurred.")
```

```
const QString SHALLOT_MUST_CLOSE_TEXT = QObject::tr("Your Shallot process is disturbed by an error and ne
```

```
const QString Value_isShallotException = "_isShallotException"
```

## 10.2.12 Namespace *sh::filesystemhandlers*

```
namespace sh::filesystemhandlers
```

Implementations of filesystem handlers.

Subclasses of *sh::filesystem::FilesystemHandler* (and possibly some auxiliary stuff). They implement how to access various types of filesystems.

Some special purpose handlers reside outside of this namespace.

```
class ActionMountGnomeIOLocation : public sh::actions::ActionActionItem
    #include <gnomeiofilesystemhandler.h> Action for mounting a GIO location.
```

## Public Functions

**ActionMountGnomeIOLocation** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>>  
nodes)

Constructed only internally.

void **execute** ()

Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \*info)

Executes this action.

QKeySequence **shortcuthint** ()

Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()

Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)

Sets the keyboard shortcut for triggering this action.

QString **text** ()

Returns the displayed text for this action.

QString **icon** ()

Returns the icon for this action.

bool **enabled** ()

Checks if this action is enabled.

bool **isChecked** ()

Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

void **setChecked** (bool *checked*)

Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)

Sets the visibility of this item.

bool **visible** ()

Checks the visibility of this item (non-recursively).

```
std::weak_ptr<AbstractActionItem> parentAction ()
    Returns the parent action, if it is added to a container.

void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.
```

## Signals

```
void changed ()
    Emits when some data changed.
```

## Public Static Functions

```
void offerMountGVolumeUuid (std::shared_ptr<sh::filesystem::FilesystemNode> node, QString
    gvolumeuuid)

void offerMountLocation (std::shared_ptr<sh::filesystem::FilesystemNode> node, QString lo-
    cation)

void unofferMount (std::shared_ptr<sh::filesystem::FilesystemNode> node)

void doInitialize ()

void doShutdown ()

QString mount (sh::actions::ActionExecutionInfo *info, QString gvolumeuuid, QString location,
    std::shared_ptr<filesystem::FilesystemNode> node)

class ActionMountNetworkGnomeIOLocation : public sh::actions::ActionActionItem
    #include <gnomeionetworkfilesystemhandler.h> Action for mounting a GIO network location.
```

## Public Functions

```
ActionMountNetworkGnomeIOLocation ()
    Constructed only internally.

void execute ()
    Executes this action.

void execute (sh::actions::ActionExecutionInfo *info)
    Executes this action.

QKeySequence shortcuthint ()
    Returns the keyboard shortcut for triggering this action.

bool shortcuthintTriggersOnCurrentDirectory ()
    Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry
    selection).
```

void **setShortcut** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<AbstractActionItem> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<AbstractActionItem> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

**class ActionUnmountGnomeIOLocation** : public *sh::actions::ActionActionItem*  
*#include <gnomeiofilesystemhandler.h>* Action for unmounting a GIO location.

## Public Functions

**ActionUnmountGnomeIOLocation** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>>  
*nodes*)

Constructed only internally.

void **execute** ()  
Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \*info)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<AbstractActionItem> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<AbstractActionItem> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
    > *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

## Public Static Functions

void **offerUnmountLocation** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> *node*, QString  
    *location*, bool *staticmode* = false)

void **unofferUnmount** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> *node*)

void **doInitialize** ()

void **doShutdown** ()

**class ArchiveFilesystemHandler** : public *sh::filesystem::FilesystemHandler*  
    #include <archivefilesystemhandler.h> A filesystem handler for archive files of some formats.

This implementation uses process calls to some external utility tools.

## Public Functions

**ArchiveFilesystemHandler** (*sh::filesystem::FilesystemModel* \**model*)

void **itemlist** (*sh::filesystem::Operation* \**op*, std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*,  
    *sh::filesystem::FilesystemNodeType* *type*, *sh::filesystem::FilesystemNodeListEditor*  
    \**list*)

*sh::filesystem::FilesystemNodeType* **getType** (*sh::filesystem::Operation* \**op*,  
    std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*)

qint64 **getSize** (*sh::filesystem::Operation* \**op*, std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*)



```

QDateTime getMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl)
void setMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
               QDateTime time)
QString getMimetype (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl)
QString getLinkTarget (sh::filesystem::Operation *op, std::shared_ptr<const
                       sh::filesystem::Eurl> eurl)
bool canGetFileContent (sh::filesystem::Operation *op, std::shared_ptr<const
                        sh::filesystem::Eurl> eurl)
std::shared_ptr<QIODevice> getFileContent (sh::filesystem::Operation *op,
                                           std::shared_ptr<const sh::filesystem::Eurl>
                                           eurl)
bool canCreateDirectory (sh::filesystem::Operation *op, std::shared_ptr<const
                          sh::filesystem::Eurl> eurl)
void createDirectory (sh::filesystem::Operation *op, std::shared_ptr<const
                       sh::filesystem::Eurl> eurl)
bool canCreateLink (sh::filesystem::Operation *op, std::shared_ptr<const
                     sh::filesystem::Eurl> eurl)
void createLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                  eurl, QString target)
bool canCreateFile (sh::filesystem::Operation *op, std::shared_ptr<const
                     sh::filesystem::Eurl> eurl)
void createFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                  eurl, QIODevice *content, HandlerTransfer *handlertransfer)
bool canDeleteItem (sh::filesystem::Operation *op, std::shared_ptr<const
                     sh::filesystem::Eurl> eurl)
void deleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                  eurl)
bool canRenameItem (sh::filesystem::Operation *op, std::shared_ptr<const
                     sh::filesystem::Eurl> src)
void renameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> src,
                  QString destpath)
QList<std::shared_ptr<sh::actions::AbstractActionItem>> getActions (const
                                                                    QList<std::shared_ptr<const
                                                                    sh::filesystem::Eurl>> eurls)

void itemlist (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
                sh::filesystem::FilesystemNodeType type, sh::filesystem::FilesystemNodeListEditor
                *list) = 0
    Determine a list of subelements in a certain directory.

void configureItems (sh::filesystem::Operation *op, QList<std::shared_ptr<FilesystemNode>>
                      nodes)
    Configure newly created nodes (e.g. setting another icon or changing the display name).

sh::filesystem::FilesystemNodeType getType (sh::filesystem::Operation *op,
                                              std::shared_ptr<const sh::filesystem::Eurl> eurl) =
    0
    Determine if a file is regular, or a dir, or a link, ...

```

qint64 **getSize** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)  
= 0

Determine the size of a file.

QDateTime **getMtime** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*>  
eurl) = 0

Determine the mtime of a file.

void **setMtime** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl,  
QDateTime time) = 0

Set the mtime of a file.

QString **getMimeType** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl*> eurl) = 0

Determine mime type of a file.

QString **getLinkTarget** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl*> eurl) = 0

Resolve a link.

QList<QString> **listExtendedAttributes** (*sh::filesystem::Operation* \*op,  
std::shared\_ptr<const *sh::filesystem::Eurl*>  
eurl)

Fetches the list of available extended attributes for an entry.

qint64 **getExtendedAttributeSize** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl*> eurl, QString attribute)

Returns the size of value for one extended attribute for an entry.

QByteArray **getExtendedAttribute** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl*> eurl, QString attribute)

Returns the value for one extended attribute for an entry.

void **setExtendedAttribute** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl*> eurl, QString attribute, QByteArray  
value)

Sets the value for one extended attribute for an entry.

void **removeExtendedAttribute** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl*> eurl, QString attribute)

Removes one extended attributes for an entry.

QMap<QString, QString> **getCustomAttributes** (*sh::filesystem::Operation*  
\*op, std::shared\_ptr<const  
*sh::filesystem::Eurl*> eurl)

Returns the custom attributes for an entry.

In contrast to extended attributes, the custom attributes are not directly stored in the filesystem this way, but handle implementation specific aspects (like permissions).

void **setCustomAttribute** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl*> eurl, QString attribute, QString value)

Sets the value for one custom attribute for an entry.

See also getCustomAttributes.

bool **canGetFileContent** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl*> eurl) = 0

Returns if it is allowed to get the content of a certain file.

std::shared\_ptr<QIODevice> **getFileContent** (*sh::filesystem::Operation* \*op,  
std::shared\_ptr<const *sh::filesystem::Eurl*>  
eurl) = 0

Get the content of a file.

```

bool canCreateDirectory (sh::filesystem::Operation *op, std::shared_ptr<const
                        sh::filesystem::Eurl> eurl) = 0
    Returns if it is allowed to create subdirectories in a certain directory.

void createDirectory (sh::filesystem::Operation *op, std::shared_ptr<const
                        sh::filesystem::Eurl> eurl) = 0
    Create a directory.

bool canCreateLink (sh::filesystem::Operation *op, std::shared_ptr<const
                        sh::filesystem::Eurl> eurl) = 0
    Returns if it is allowed to create a link in a certain directory.

void createLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                  eurl, QString target) = 0
    Create a link.

bool canCreateFile (sh::filesystem::Operation *op, std::shared_ptr<const
                        sh::filesystem::Eurl> eurl) = 0
    Returns if it is allowed to create files in a certain directory.

void createFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                  eurl, QIODevice *content, HandlerTransfer *handlertransfer) = 0
    Creates a file with some content.

    It may use handlertransfer for some better integration, like cancel support and progress moni-
    toring.

bool canDeleteItem (sh::filesystem::Operation *op, std::shared_ptr<const
                        sh::filesystem::Eurl> eurl) = 0
    Returns if it is allowed to delete a certain file/directory/link/...

void deleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                  eurl) = 0
    Delete a file/directory/link/...

void deleteDirectoryIfEmpty (sh::filesystem::Operation *op, std::shared_ptr<const
                        sh::filesystem::Eurl> eurl)
    Deletes a directory only if it is empty.

bool canRenameItem (sh::filesystem::Operation *op, std::shared_ptr<const
                        sh::filesystem::Eurl> src) = 0
    Returns if it is allowed to rename a certain file, directory, link, ... (see renameItem).

void renameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> src,
                  QString destpath) = 0
    Renames an item to another path.

    It can be a simple filename change or also changes in the deeper path segments.

QList<std::shared_ptr<sh::actions::AbstractActionItem>> getActions (const
                                                                QList<std::shared_ptr<const
                                                                sh::filesystem::Eurl>> eu-
                                                                rls) = 0
    Returns a list of actions (see former example) for showing for certain files.

bool requiresResolvedLinks ()
    Returns if this handler requires to be used by the engine with resolved links.

void viewEnteredDirectory (std::shared_ptr<const sh::filesystem::Eurl>)
    Callback for preparing stuff when the view entered some directory (e.g. for enabling watchers). See
    also viewLeftDirectory.

void viewLeftDirectory (std::shared_ptr<const sh::filesystem::Eurl>)
    Callback for preparing stuff when the view left some directory. See also viewEnteredDirectory.

```

bool **isWellKnownScheme** ()

Returns if the scheme for this handler is a well known one (which external programs typically would understand).

void **search** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl*> eurl, std::function<void> std::shared\_ptr<const  
*sh::filesystem::Eurl*>, QList<std::shared\_ptr<*sh::search::SearchCriterion*>>,  
*sh::filesystem::FilesystemNodeType* ftype  
> addfile, std::function<void>std::shared\_ptr<const *sh::filesystem::Eurl*>> visitdir,  
QList<std::shared\_ptr<*sh::search::SearchCriterion*>> criteria) Helps searching for files.

Override this method with a behavior identical to the default, if a specialized implementation can be much faster than the default one.

void **customizeUi** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> node, bool \*showSearch-  
Panel)

Creates a custom gui, which becomes part of the fileview.

*sh::filesystem::FilesystemModel* \***model** ()

A convenience shortcut to the *sh::filesystem::FilesystemModel* (a singleton).

## Public Static Functions

void **doInitialize** ()

void **doShutdown** ()

**class GnomeIODevicesFilesystemHandler** : public *sh::filesystemhandlers::GnomeIOFilesystemHandler*, public  
#include <gnomeiodevicesfilesystemhandler.h> A filesystem handler for the GIO filesystem 'Devices'  
subtree.

## Public Functions

**~GnomeIODevicesFilesystemHandler** ()

void **itemlist** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl,  
*sh::filesystem::FilesystemNodeType* type, *sh::filesystem::FilesystemNodeListEditor*  
\*list) **override**

*sh::filesystem::FilesystemNodeType* **getType** (*sh::filesystem::Operation* \*op,  
std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)  
**override**

QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> **getActions** (const  
QList<std::shared\_ptr<const  
*sh::filesystem::Eurl*>> eu-  
rls) **override**

void **refresh** ()  
Refreshes the device list.

void **requestNewNode** (std::shared\_ptr<const *sh::filesystem::Eurl*> eurl,  
std::shared\_ptr<*sh::filesystem::FilesystemNode*> \*nnode)  
Requests a new device node.

void **itemlist** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl,  
*sh::filesystem::FilesystemNodeType* type, *sh::filesystem::FilesystemNodeListEditor*  
\*list) = 0  
Determine a list of subelements in a certain directory.

```

sh::filesystem::FileType getType (sh::filesystem::Operation *op,
                                   std::shared_ptr<const sh::filesystem::Eurl> eurl) =
                                   0
    Determine if a file is regular, or a dir, or a link, ...

qint64 getSize (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl)
qint64 getSize (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl)
                                   = 0
    Determine the size of a file.

QDateTime getMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl)
QDateTime getMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl) = 0
    Determine the mtime of a file.

void setMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
               QDateTime time)
void setMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
               QDateTime time) = 0
    Set the mtime of a file.

QString getMimeType (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl)
QString getMimeType (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) = 0
    Determine mime type of a file.

QString getLinkTarget (sh::filesystem::Operation *op, std::shared_ptr<const
                      sh::filesystem::Eurl> eurl)
QString getLinkTarget (sh::filesystem::Operation *op, std::shared_ptr<const
                      sh::filesystem::Eurl> eurl) = 0
    Resolve a link.

bool canGetFileContent (sh::filesystem::Operation *op, std::shared_ptr<const
                      sh::filesystem::Eurl> eurl)
bool canGetFileContent (sh::filesystem::Operation *op, std::shared_ptr<const
                      sh::filesystem::Eurl> eurl) = 0
    Returns if it is allowed to get the content of a certain file.

std::shared_ptr<QIODevice> getFileContent (sh::filesystem::Operation *op,
                                           std::shared_ptr<const sh::filesystem::Eurl>
                                           eurl)
std::shared_ptr<QIODevice> getFileContent (sh::filesystem::Operation *op,
                                           std::shared_ptr<const sh::filesystem::Eurl>
                                           eurl) = 0
    Get the content of a file.

bool canCreateDirectory (sh::filesystem::Operation *op, std::shared_ptr<const
                      sh::filesystem::Eurl> eurl)
bool canCreateDirectory (sh::filesystem::Operation *op, std::shared_ptr<const
                      sh::filesystem::Eurl> eurl) = 0
    Returns if it is allowed to create subdirectories in a certain directory.

void createDirectory (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl)

```

```
void createDirectory (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) = 0
    Create a directory.

bool canCreateLink (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl)

bool canCreateLink (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) = 0
    Returns if it is allowed to create a link in a certain directory.

void createLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                eurl, QString target)

void createLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                eurl, QString target) = 0
    Create a link.

bool canCreateFile (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl)

bool canCreateFile (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) = 0
    Returns if it is allowed to create files in a certain directory.

void createFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                eurl, QIODevice *content, HandlerTransfer *handlertransfer)

void createFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                eurl, QIODevice *content, HandlerTransfer *handlertransfer) = 0
    Creates a file with some content.

    It may use handlertransfer for some better integration, like cancel support and progress moni-
    toring.

bool canDeleteItem (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl)

bool canDeleteItem (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) = 0
    Returns if it is allowed to delete a certain file/directory/link/...

void deleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                eurl)

void deleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                eurl) = 0
    Delete a file/directory/link/...

bool canRenameItem (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> src)

bool canRenameItem (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> src) = 0
    Returns if it is allowed to rename a certain file, directory, link, ... (see renameItem).

void renameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> src,
                QString destpath)

void renameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> src,
                QString destpath) = 0
    Renames an item to another path.

    It can be a simple filename change or also changes in the deeper path segments.
```



`QList<std::shared_ptr<sh::actions::AbstractActionItem>> getActions (const  
QList<std::shared_ptr<const  
sh::filesystem::Eurl>> eu-  
rls) = 0`

Returns a list of actions (see former example) for showing for certain files.

`bool requiresResolvedLinks ()`

Returns if this handler requires to be used by the engine with resolved links.

`bool isWellKnownScheme ()`

Returns if the scheme for this handler is a well known one (which external programs typically would understand).

`void configureItems (sh::filesystem::Operation *op, QList<std::shared_ptr<FilesystemNode>>  
nodes)`

Configure newly created nodes (e.g. setting another icon or changing the display name).

`QList<QString> listExtendedAttributes (sh::filesystem::Operation *op,  
std::shared_ptr<const sh::filesystem::Eurl>  
eurl)`

Fetches the list of available extended attributes for an entry.

`quint64 getExtendedAttributeSize (sh::filesystem::Operation *op, std::shared_ptr<const  
sh::filesystem::Eurl> eurl, QString attribute)`

Returns the size of value for one extended attribute for an entry.

`QByteArray getExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const  
sh::filesystem::Eurl> eurl, QString attribute)`

Returns the value for one extended attribute for an entry.

`void setExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const  
sh::filesystem::Eurl> eurl, QString attribute, QByteArray  
value)`

Sets the value for one extended attribute for an entry.

`void removeExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const  
sh::filesystem::Eurl> eurl, QString attribute)`

Removes one extended attributes for an entry.

`QMap<QString, QString> getCustomAttributes (sh::filesystem::Operation  
*op, std::shared_ptr<const  
sh::filesystem::Eurl> eurl)`

Returns the custom attributes for an entry.

In contrast to extended attributes, the custom attributes are not directly stored in the filesystem this way, but handle implementation specific aspects (like permissions).

`void setCustomAttribute (sh::filesystem::Operation *op, std::shared_ptr<const  
sh::filesystem::Eurl> eurl, QString attribute, QString value)`

Sets the value for one custom attribute for an entry.

See also `getCustomAttributes`.

`void deleteDirectoryIfEmpty (sh::filesystem::Operation *op, std::shared_ptr<const  
sh::filesystem::Eurl> eurl)`

Deletes a directory only if it is empty.

`void viewEnteredDirectory (std::shared_ptr<const sh::filesystem::Eurl>)`

Callback for preparing stuff when the view entered some directory (e.g. for enabling watchers). See also `viewLeftDirectory`.

`void viewLeftDirectory (std::shared_ptr<const sh::filesystem::Eurl>)`

Callback for preparing stuff when the view left some directory. See also `viewEnteredDirectory`.

```
void search (sh::filesystem::Operation *op, std::shared_ptr<const  
    sh::filesystem::Eurl> eurl, std::function<void> std::shared_ptr<const  
    sh::filesystem::Eurl>, QList<std::shared_ptr<sh::search::SearchCriterion>>,  
    sh::filesystem::FilesystemNodeType ftype  
> addfile, std::function<void>std::shared_ptr<const sh::filesystem::Eurl>> visitdir,  
    QList<std::shared_ptr<sh::search::SearchCriterion>> criteria)Helps searching for files.
```

Override this method with a behavior identical to the default, if a specialized implementation can be much faster than the default one.

```
void customizeUi (std::shared_ptr<sh::filesystem::FilesystemNode> node, bool *showSearch-  
    Panel)
```

Creates a custom gui, which becomes part of the fileview.

```
sh::filesystem::FilesystemModel *model ()
```

A convenience shortcut to the *sh::filesystem::FilesystemModel* (a singleton).

```
void doShutdown ()
```

Executes singleton shutdown.

```
void shutdown ()
```

Shutdown down this singleton.

```
bool isAlive ()
```

Returns if this singleton is alive (true until its shutdown begins).

## Public Static Functions

```
QString mountDevice (QString gvolumeuuid, sh::actions::ActionExecutionInfo *info)
```

```
QString mountLocation (QString gurl, sh::actions::ActionExecutionInfo *info)
```

```
QString unmountLocation (QString gurl, sh::actions::ActionExecutionInfo *info)
```

```
QByteArray eurl2gurlb (std::shared_ptr<const sh::filesystem::Eurl> eurl)
```

```
QString eurl2gurl (std::shared_ptr<const sh::filesystem::Eurl> eurl)
```

```
std::shared_ptr<const sh::filesystem::Eurl> gurl2eurl (QString gurl)
```

```
void logIfError (void *error)
```

```
void throwIfError (void *error)
```

## Private Functions

```
GnomeIODevicesFilesystemHandler ()
```

## Friends

```
friend class ActionMount
```

```
class GnomeIOFilesystemHandler : public sh::filesystem::FilesystemHandler  
    #include <gnomeiofilesystemhandler.h> A filesystem handler for GIO filesystem parts.
```

This is the abstract base class.

Subclassed by *sh::filesystemhandlers::GnomeIODevicesFilesystemHandler*,  
*sh::filesystemhandlers::GnomeIOGenericFilesystemHandler*, *sh::filesystemhandlers::GnomeIONetworkFilesystemHandler*,  
*sh::filesystemhandlers::GnomeIOSmbFilesystemHandler*



## Public Functions

**GnomeIOFilesystemHandler()**

void **itemlist** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, *sh::filesystem::FilesystemNodeType* type, *sh::filesystem::FilesystemNodeListEditor* \*list)

*sh::filesystem::FilesystemNodeType* **getType** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)

qint64 **getSize** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)

QDateTime **getMtime** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)

void **setMtime** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QDateTime time)

QString **getMimeType** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)

QString **getLinkTarget** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)

bool **canGetFileContent** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)

std::shared\_ptr<QIODevice> **getFileContent** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)

bool **canCreateDirectory** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)

void **createDirectory** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)

bool **canCreateLink** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)

void **createLink** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QString target)

bool **canCreateFile** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)

void **createFile** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QIODevice \*content, HandlerTransfer \*handlertransfer)

bool **canDeleteItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)

void **deleteItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)

bool **canRenameItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> src)

void **renameItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> src, QString destpath)

QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> **getActions** (const QList<std::shared\_ptr<const *sh::filesystem::Eurl*>> eurls)

**bool requiresResolvedLinks ()**  
Returns if this handler requires to be used by the engine with resolved links.

**bool isWellKnownScheme ()**  
Returns if the scheme for this handler is a well known one (which external programs typically would understand).

**void itemList (sh::filesystem::Operation \*op, std::shared\_ptr<const sh::filesystem::Eurl> eurl, sh::filesystem::FilesystemNodeType type, sh::filesystem::FilesystemNodeListEditor \*list) = 0**  
Determine a list of subelements in a certain directory.

**void configureItems (sh::filesystem::Operation \*op, QList<std::shared\_ptr<FilesystemNode>> nodes)**  
Configure newly created nodes (e.g. setting another icon or changing the display name).

**sh::filesystem::FilesystemNodeType getType (sh::filesystem::Operation \*op, std::shared\_ptr<const sh::filesystem::Eurl> eurl) = 0**  
Determine if a file is regular, or a dir, or a link, ...

**qint64 getSize (sh::filesystem::Operation \*op, std::shared\_ptr<const sh::filesystem::Eurl> eurl) = 0**  
Determine the size of a file.

**QDateTime getMtime (sh::filesystem::Operation \*op, std::shared\_ptr<const sh::filesystem::Eurl> eurl) = 0**  
Determine the mtime of a file.

**void setMtime (sh::filesystem::Operation \*op, std::shared\_ptr<const sh::filesystem::Eurl> eurl, QDateTime time) = 0**  
Set the mtime of a file.

**QString getMimeType (sh::filesystem::Operation \*op, std::shared\_ptr<const sh::filesystem::Eurl> eurl) = 0**  
Determine mime type of a file.

**QString getLinkTarget (sh::filesystem::Operation \*op, std::shared\_ptr<const sh::filesystem::Eurl> eurl) = 0**  
Resolve a link.

**QList<QString> listExtendedAttributes (sh::filesystem::Operation \*op, std::shared\_ptr<const sh::filesystem::Eurl> eurl)**  
Fetches the list of available extended attributes for an entry.

**qint64 getExtendedAttributeSize (sh::filesystem::Operation \*op, std::shared\_ptr<const sh::filesystem::Eurl> eurl, QString attribute)**  
Returns the size of value for one extended attribute for an entry.

**QByteArray getExtendedAttribute (sh::filesystem::Operation \*op, std::shared\_ptr<const sh::filesystem::Eurl> eurl, QString attribute)**  
Returns the value for one extended attribute for an entry.

**void setExtendedAttribute (sh::filesystem::Operation \*op, std::shared\_ptr<const sh::filesystem::Eurl> eurl, QString attribute, QByteArray value)**  
Sets the value for one extended attribute for an entry.

**void removeExtendedAttribute (sh::filesystem::Operation \*op, std::shared\_ptr<const sh::filesystem::Eurl> eurl, QString attribute)**  
Removes one extended attributes for an entry.

QMap<QString, QString> **getCustomAttributes** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)

Returns the custom attributes for an entry.

In contrast to extended attributes, the custom attributes are not directly stored in the filesystem this way, but handle implementation specific aspects (like permissions).

void **setCustomAttribute** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QString attribute, QString value)

Sets the value for one custom attribute for an entry.

See also getCustomAttributes.

bool **canGetFileContent** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0

Returns if it is allowed to get the content of a certain file.

std::shared\_ptr<QIODevice> **getFileContent** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0

Get the content of a file.

bool **canCreateDirectory** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0

Returns if it is allowed to create subdirectories in a certain directory.

void **createDirectory** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0

Create a directory.

bool **canCreateLink** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0

Returns if it is allowed to create a link in a certain directory.

void **createLink** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QString target) = 0

Create a link.

bool **canCreateFile** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0

Returns if it is allowed to create files in a certain directory.

void **createFile** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QIODevice \*content, HandlerTransfer \*handlertransfer) = 0

Creates a file with some content.

It may use handlertransfer for some better integration, like cancel support and progress monitoring.

bool **canDeleteItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0

Returns if it is allowed to delete a certain file/directory/link/...

void **deleteItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0

Delete a file/directory/link/...

void **deleteDirectoryIfEmpty** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)

Deletes a directory only if it is empty.

bool **canRenameItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> src) = 0

Returns if it is allowed to rename a certain file, directory, link, ... (see renameItem).

void **renameItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> src, QString destpath) = 0  
Renames an item to another path.

It can be a simple filename change or also changes in the deeper path segments.

QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> **getActions** (const QList<std::shared\_ptr<const *sh::filesystem::Eurl*>> eurls) = 0

Returns a list of actions (see former example) for showing for certain files.

void **viewEnteredDirectory** (std::shared\_ptr<const *sh::filesystem::Eurl*>)  
Callback for preparing stuff when the view entered some directory (e.g. for enabling watchers). See also viewLeftDirectory.

void **viewLeftDirectory** (std::shared\_ptr<const *sh::filesystem::Eurl*>)  
Callback for preparing stuff when the view left some directory. See also viewEnteredDirectory.

void **search** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, std::function<void> std::shared\_ptr<const *sh::filesystem::Eurl*>, QList<std::shared\_ptr<*sh::search::SearchCriterion*>>, *sh::filesystem::FilesystemNodeType* ftype  
> addfile, std::function<void>std::shared\_ptr<const *sh::filesystem::Eurl*>> visitdir, QList<std::shared\_ptr<*sh::search::SearchCriterion*>> criteriaHelps searching for files.

Override this method with a behavior identical to the default, if a specialized implementation can be much faster than the default one.

void **customizeUi** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> node, bool \*showSearchPanel)  
Creates a custom gui, which becomes part of the fileview.

*sh::filesystem::FilesystemModel* \***model** ()  
A convenience shortcut to the *sh::filesystem::FilesystemModel* (a singleton).

## Public Static Functions

QString **mountDevice** (QString gvolumeuuid, *sh::actions::ActionExecutionInfo* \*info)

QString **mountLocation** (QString gurl, *sh::actions::ActionExecutionInfo* \*info)

QString **unmountLocation** (QString gurl, *sh::actions::ActionExecutionInfo* \*info)

QByteArray **eurl2gurlb** (std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)

QString **eurl2gurl** (std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)

std::shared\_ptr<const *sh::filesystem::Eurl*> **gurl2eurl** (QString gurl)

void **logIfError** (void \*error)

void **throwIfError** (void \*error)

**class MountOperations**  
#include <gnoimeiofilesystemhandler.h>

## Public Static Functions

```
void *createMountOperation (sh::actions::ActionExecutionInfo *info, QString address,
                           MountOperationsCallbackData **opdata)
void deleteMountOperation (MountOperationsCallbackData *opdata)
```

## Public Static Attributes

```
QHash<size_t, MountOperationsCallbackData*> callbackdata
size_t callbackdataindex
QMutex callbackdatamutex
struct MountOperationsCallbackData
#include <gnomeiofilesystemhandler.h>
```

## Public Functions

```
MountOperationsCallbackData (sh::actions::ActionExecutionInfo *info, size_t index,
                             void *mountoperation, QString address)
```

## Public Members

```
sh::actions::ActionExecutionInfo *info
size_t index
void *mountoperation
QString address
class GnomeIOGenericFilesystemHandler : public sh::filesystemhandlers::GnomeIOFilesystemHandler
#include <gnomeiogenericfilesystemhandler.h> A filesystem handler for common/unspecial GIO filesystem parts.
```

This is a generic non-abstract implementation without special behavior.

## Public Functions

```
GnomeIOGenericFilesystemHandler (sh::filesystem::FilesystemModel *model)
void itemList (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
               sh::filesystem::FilesystemNodeType type, sh::filesystem::FilesystemNodeListEditor
               *list)
void itemList (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
               sh::filesystem::FilesystemNodeType type, sh::filesystem::FilesystemNodeListEditor
               *list) = 0
    Determine a list of subelements in a certain directory.
sh::filesystem::FilesystemNodeType getType (sh::filesystem::Operation *op,
                                             std::shared_ptr<const sh::filesystem::Eurl> eurl)
sh::filesystem::FilesystemNodeType getType (sh::filesystem::Operation *op,
                                             std::shared_ptr<const sh::filesystem::Eurl> eurl) =
0
    Determine if a file is regular, or a dir, or a link, ...
```

qint64 **getSize** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)  
qint64 **getSize** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)  
= 0  
Determine the size of a file.

QDateTime **getMtime** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)  
QDateTime **getMtime** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0  
Determine the mtime of a file.

void **setMtime** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QDateTime time)  
void **setMtime** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QDateTime time) = 0  
Set the mtime of a file.

QString **getMimeType** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)  
QString **getMimeType** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0  
Determine mime type of a file.

QString **getLinkTarget** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)  
QString **getLinkTarget** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0  
Resolve a link.

bool **canGetFileContent** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)  
bool **canGetFileContent** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0  
Returns if it is allowed to get the content of a certain file.

std::shared\_ptr<QIODevice> **getFileContent** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)  
std::shared\_ptr<QIODevice> **getFileContent** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0  
Get the content of a file.

bool **canCreateDirectory** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)  
bool **canCreateDirectory** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0  
Returns if it is allowed to create subdirectories in a certain directory.

void **createDirectory** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)  
void **createDirectory** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0  
Create a directory.

bool **canCreateLink** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)



```

bool canCreateLink (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) = 0
    Returns if it is allowed to create a link in a certain directory.

void createLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                eurl, QString target)

void createLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                eurl, QString target) = 0
    Create a link.

bool canCreateFile (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl)

bool canCreateFile (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) = 0
    Returns if it is allowed to create files in a certain directory.

void createFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                eurl, QIODevice *content, HandlerTransfer *handlertransfer)

void createFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                eurl, QIODevice *content, HandlerTransfer *handlertransfer) = 0
    Creates a file with some content.

    It may use handlertransfer for some better integration, like cancel support and progress moni-
    toring.

bool canDeleteItem (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl)

bool canDeleteItem (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) = 0
    Returns if it is allowed to delete a certain file/directory/link/...

void deleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                eurl)

void deleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                eurl) = 0
    Delete a file/directory/link/...

bool canRenameItem (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> src)

bool canRenameItem (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> src) = 0
    Returns if it is allowed to rename a certain file, directory, link, ... (see renameItem).

void renameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> src,
                QString destpath)

void renameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> src,
                QString destpath) = 0
    Renames an item to another path.

    It can be a simple filename change or also changes in the deeper path segments.

QList<std::shared_ptr<sh::actions::AbstractActionItem>> getActions (const
                                                                QList<std::shared_ptr<const
                                                                sh::filesystem::Eurl>> eu-
                                                                rls)

```

`QList<std::shared_ptr<sh::actions::AbstractActionItem>> getActions (const`  
`QList<std::shared_ptr<const`  
`sh::filesystem::Eurl>> eurl`  
`rls) = 0`  
Returns a list of actions (see former example) for showing for certain files.

`bool requiresResolvedLinks ()`  
Returns if this handler requires to be used by the engine with resolved links.

`bool isWellKnownScheme ()`  
Returns if the scheme for this handler is a well known one (which external programs typically would understand).

`void configureItems (sh::filesystem::Operation *op, QList<std::shared_ptr<FilesystemNode>>`  
`nodes)`  
Configure newly created nodes (e.g. setting another icon or changing the display name).

`QList<QString> listExtendedAttributes (sh::filesystem::Operation *op,`  
`std::shared_ptr<const sh::filesystem::Eurl>`  
`eurl)`  
Fetches the list of available extended attributes for an entry.

`quint64 getExtendedAttributeSize (sh::filesystem::Operation *op, std::shared_ptr<const`  
`sh::filesystem::Eurl> eurl, QString attribute)`  
Returns the size of value for one extended attribute for an entry.

`QByteArray getExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const`  
`sh::filesystem::Eurl> eurl, QString attribute)`  
Returns the value for one extended attribute for an entry.

`void setExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const`  
`sh::filesystem::Eurl> eurl, QString attribute, QByteArray`  
`value)`  
Sets the value for one extended attribute for an entry.

`void removeExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const`  
`sh::filesystem::Eurl> eurl, QString attribute)`  
Removes one extended attributes for an entry.

`QMap<QString, QString> getCustomAttributes (sh::filesystem::Operation`  
`*op, std::shared_ptr<const`  
`sh::filesystem::Eurl> eurl)`  
Returns the custom attributes for an entry.  
  
In contrast to extended attributes, the custom attributes are not directly stored in the filesystem this way, but handle implementation specific aspects (like permissions).

`void setCustomAttribute (sh::filesystem::Operation *op, std::shared_ptr<const`  
`sh::filesystem::Eurl> eurl, QString attribute, QString value)`  
Sets the value for one custom attribute for an entry.  
  
See also `getCustomAttributes`.

`void deleteDirectoryIfEmpty (sh::filesystem::Operation *op, std::shared_ptr<const`  
`sh::filesystem::Eurl> eurl)`  
Deletes a directory only if it is empty.

`void viewEnteredDirectory (std::shared_ptr<const sh::filesystem::Eurl>)`  
Callback for preparing stuff when the view entered some directory (e.g. for enabling watchers). See also `viewLeftDirectory`.

`void viewLeftDirectory (std::shared_ptr<const sh::filesystem::Eurl>)`  
Callback for preparing stuff when the view left some directory. See also `viewEnteredDirectory`.



```
void search (sh::filesystem::Operation *op, std::shared_ptr<const
             sh::filesystem::Eurl> eurl, std::function<void> std::shared_ptr<const
             sh::filesystem::Eurl>, QList<std::shared_ptr<sh::search::SearchCriterion>>,
             sh::filesystem::FilesystemNodeType ftype
             > addfile, std::function<void>std::shared_ptr<const sh::filesystem::Eurl>> visitdir,
             QList<std::shared_ptr<sh::search::SearchCriterion>> criteriaHelps searching for files.
```

Override this method with a behavior identical to the default, if a specialized implementation can be much faster than the default one.

```
void customizeUi (std::shared_ptr<sh::filesystem::FilesystemNode> node, bool *showSearch-
                  Panel)
```

Creates a custom gui, which becomes part of the fileview.

```
sh::filesystem::FilesystemModel *model ()
```

A convenience shortcut to the *sh::filesystem::FilesystemModel* (a singleton).

## Public Static Functions

```
QString mountDevice (QString gvolumeuuid, sh::actions::ActionExecutionInfo *info)
```

```
QString mountLocation (QString gurl, sh::actions::ActionExecutionInfo *info)
```

```
QString unmountLocation (QString gurl, sh::actions::ActionExecutionInfo *info)
```

```
QByteArray eurl2gurlb (std::shared_ptr<const sh::filesystem::Eurl> eurl)
```

```
QString eurl2gurl (std::shared_ptr<const sh::filesystem::Eurl> eurl)
```

```
std::shared_ptr<const sh::filesystem::Eurl> gurl2eurl (QString gurl)
```

```
void logIfError (void *error)
```

```
void throwIfError (void *error)
```

```
class GnomeIONetworkFilesystemHandler : public sh::filesystemhandlers::GnomeIOFilesystemHandler, public
    #include <gnomeionetworkfilesystemhandler.h> A filesystem handler for the GIO filesystem 'Network'
    subtree.
```

## Public Functions

```
QList<std::shared_ptr<sh::actions::AbstractActionItem>> getActions (const
                                                                    QList<std::shared_ptr<const
                                                                    sh::filesystem::Eurl>> eu-
                                                                    rls) override
```

```
void requestNewNode (std::shared_ptr<const sh::filesystem::Eurl> eurl,
                    std::shared_ptr<sh::filesystem::FilesystemNode> *nnode)
    Requests a new network node.
```

```
void itemlist (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
               sh::filesystem::FilesystemNodeType type, sh::filesystem::FilesystemNodeListEditor
               *list)
```

```
void itemlist (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
               sh::filesystem::FilesystemNodeType type, sh::filesystem::FilesystemNodeListEditor
               *list) = 0
    Determine a list of subelements in a certain directory.
```

```
sh::filesystem::FilesystemNodeType getType (sh::filesystem::Operation *op,
                                              std::shared_ptr<const sh::filesystem::Eurl> eurl)
```

*sh::filesystem::FileSystemNodeType* **getType** (*sh::filesystem::Operation* \**op*,  
std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*) =  
0  
Determine if a file is regular, or a dir, or a link, ...

qint64 **getSize** (*sh::filesystem::Operation* \**op*, std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*)  
qint64 **getSize** (*sh::filesystem::Operation* \**op*, std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*)  
= 0  
Determine the size of a file.

QDateTime **getMtime** (*sh::filesystem::Operation* \**op*, std::shared\_ptr<const *sh::filesystem::Eurl*>  
*eurl*)  
QDateTime **getMtime** (*sh::filesystem::Operation* \**op*, std::shared\_ptr<const *sh::filesystem::Eurl*>  
*eurl*) = 0  
Determine the mtime of a file.

void **setMtime** (*sh::filesystem::Operation* \**op*, std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*,  
QDateTime *time*)  
void **setMtime** (*sh::filesystem::Operation* \**op*, std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*,  
QDateTime *time*) = 0  
Set the mtime of a file.

QString **getMimeType** (*sh::filesystem::Operation* \**op*, std::shared\_ptr<const  
*sh::filesystem::Eurl*> *eurl*)  
QString **getMimeType** (*sh::filesystem::Operation* \**op*, std::shared\_ptr<const  
*sh::filesystem::Eurl*> *eurl*) = 0  
Determine mime type of a file.

QString **getLinkTarget** (*sh::filesystem::Operation* \**op*, std::shared\_ptr<const  
*sh::filesystem::Eurl*> *eurl*)  
QString **getLinkTarget** (*sh::filesystem::Operation* \**op*, std::shared\_ptr<const  
*sh::filesystem::Eurl*> *eurl*) = 0  
Resolve a link.

bool **canGetFileContent** (*sh::filesystem::Operation* \**op*, std::shared\_ptr<const  
*sh::filesystem::Eurl*> *eurl*)  
bool **canGetFileContent** (*sh::filesystem::Operation* \**op*, std::shared\_ptr<const  
*sh::filesystem::Eurl*> *eurl*) = 0  
Returns if it is allowed to get the content of a certain file.

std::shared\_ptr<QIODevice> **getFileContent** (*sh::filesystem::Operation* \**op*,  
std::shared\_ptr<const *sh::filesystem::Eurl*>  
*eurl*)  
std::shared\_ptr<QIODevice> **getFileContent** (*sh::filesystem::Operation* \**op*,  
std::shared\_ptr<const *sh::filesystem::Eurl*>  
*eurl*) = 0  
Get the content of a file.

bool **canCreateDirectory** (*sh::filesystem::Operation* \**op*, std::shared\_ptr<const  
*sh::filesystem::Eurl*> *eurl*)  
bool **canCreateDirectory** (*sh::filesystem::Operation* \**op*, std::shared\_ptr<const  
*sh::filesystem::Eurl*> *eurl*) = 0  
Returns if it is allowed to create subdirectories in a certain directory.

void **createDirectory** (*sh::filesystem::Operation* \**op*, std::shared\_ptr<const  
*sh::filesystem::Eurl*> *eurl*)

```

void createDirectory (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) = 0
    Create a directory.

bool canCreateLink (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl)

bool canCreateLink (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) = 0
    Returns if it is allowed to create a link in a certain directory.

void createLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                eurl, QString target)

void createLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                eurl, QString target) = 0
    Create a link.

bool canCreateFile (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl)

bool canCreateFile (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) = 0
    Returns if it is allowed to create files in a certain directory.

void createFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                eurl, QIODevice *content, HandlerTransfer *handlertransfer)

void createFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                eurl, QIODevice *content, HandlerTransfer *handlertransfer) = 0
    Creates a file with some content.

    It may use handlertransfer for some better integration, like cancel support and progress moni-
    toring.

bool canDeleteItem (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl)

bool canDeleteItem (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) = 0
    Returns if it is allowed to delete a certain file/directory/link/...

void deleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                eurl)

void deleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                eurl) = 0
    Delete a file/directory/link/...

bool canRenameItem (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> src)

bool canRenameItem (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> src) = 0
    Returns if it is allowed to rename a certain file, directory, link, ... (see renameItem).

void renameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> src,
                QString destpath)

void renameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> src,
                QString destpath) = 0
    Renames an item to another path.

    It can be a simple filename change or also changes in the deeper path segments.

```

`QList<std::shared_ptr<sh::actions::AbstractActionItem>> getActions (const`  
`QList<std::shared_ptr<const`  
`sh::filesystem::Eurl>> eurl`  
`rls) = 0`  
Returns a list of actions (see former example) for showing for certain files.

`bool requiresResolvedLinks ()`  
Returns if this handler requires to be used by the engine with resolved links.

`bool isWellKnownScheme ()`  
Returns if the scheme for this handler is a well known one (which external programs typically would understand).

`void configureItems (sh::filesystem::Operation *op, QList<std::shared_ptr<FilesystemNode>>`  
`nodes)`  
Configure newly created nodes (e.g. setting another icon or changing the display name).

`QList<QString> listExtendedAttributes (sh::filesystem::Operation *op,`  
`std::shared_ptr<const sh::filesystem::Eurl>`  
`eurl)`  
Fetches the list of available extended attributes for an entry.

`quint64 getExtendedAttributeSize (sh::filesystem::Operation *op, std::shared_ptr<const`  
`sh::filesystem::Eurl> eurl, QString attribute)`  
Returns the size of value for one extended attribute for an entry.

`QByteArray getExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const`  
`sh::filesystem::Eurl> eurl, QString attribute)`  
Returns the value for one extended attribute for an entry.

`void setExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const`  
`sh::filesystem::Eurl> eurl, QString attribute, QByteArray`  
`value)`  
Sets the value for one extended attribute for an entry.

`void removeExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const`  
`sh::filesystem::Eurl> eurl, QString attribute)`  
Removes one extended attributes for an entry.

`QMap<QString, QString> getCustomAttributes (sh::filesystem::Operation`  
`*op, std::shared_ptr<const`  
`sh::filesystem::Eurl> eurl)`  
Returns the custom attributes for an entry.  
  
In contrast to extended attributes, the custom attributes are not directly stored in the filesystem this way, but handle implementation specific aspects (like permissions).

`void setCustomAttribute (sh::filesystem::Operation *op, std::shared_ptr<const`  
`sh::filesystem::Eurl> eurl, QString attribute, QString value)`  
Sets the value for one custom attribute for an entry.  
  
See also `getCustomAttributes`.

`void deleteDirectoryIfEmpty (sh::filesystem::Operation *op, std::shared_ptr<const`  
`sh::filesystem::Eurl> eurl)`  
Deletes a directory only if it is empty.

`void viewEnteredDirectory (std::shared_ptr<const sh::filesystem::Eurl>)`  
Callback for preparing stuff when the view entered some directory (e.g. for enabling watchers). See also `viewLeftDirectory`.

`void viewLeftDirectory (std::shared_ptr<const sh::filesystem::Eurl>)`  
Callback for preparing stuff when the view left some directory. See also `viewEnteredDirectory`.

```
void search (sh::filesystem::Operation *op, std::shared_ptr<const
sh::filesystem::Eurl> eurl, std::function<void> std::shared_ptr<const
sh::filesystem::Eurl>, QList<std::shared_ptr<sh::search::SearchCriterion>>,
sh::filesystem::FilesystemNodeType ftype
> addfile, std::function<void>std::shared_ptr<const sh::filesystem::Eurl>> visitdir,
QList<std::shared_ptr<sh::search::SearchCriterion>> criteria)
```

Helps searching for files.

Override this method with a behavior identical to the default, if a specialized implementation can be much faster than the default one.

```
void customizeUi (std::shared_ptr<sh::filesystem::FilesystemNode> node, bool *showSearch-
Panel)
```

Creates a custom gui, which becomes part of the fileview.

```
sh::filesystem::FilesystemModel *model ()
```

A convenience shortcut to the *sh::filesystem::FilesystemModel* (a singleton).

```
void doShutdown ()
```

Executes singleton shutdown.

```
void shutdown ()
```

Shutdown down this singleton.

```
bool isAlive ()
```

Returns if this singleton is alive (true until its shutdown begins).

## Public Static Functions

```
QString mountDevice (QString gvolumeuuid, sh::actions::ActionExecutionInfo *info)
```

```
QString mountLocation (QString gurl, sh::actions::ActionExecutionInfo *info)
```

```
QString unmountLocation (QString gurl, sh::actions::ActionExecutionInfo *info)
```

```
QByteArray eurl2gurlb (std::shared_ptr<const sh::filesystem::Eurl> eurl)
```

```
QString eurl2gurl (std::shared_ptr<const sh::filesystem::Eurl> eurl)
```

```
std::shared_ptr<const sh::filesystem::Eurl> gurl2eurl (QString gurl)
```

```
void logIfError (void *error)
```

```
void throwIfError (void *error)
```

## Private Functions

```
GnomeIONetworkFilesystemHandler ()
```

```
class GnomeIOSmbFilesystemHandler : public sh::filesystemhandlers::GnomeIOFilesystemHandler, public sh::b
#include <gnomeiosmbfilesystemhandler.h> A filesystem handler for the GIO filesystem 'smb' subtrees
(in 'Network').
```

## Public Functions

```
sh::filesystem::FilesystemNodeType getType (sh::filesystem::Operation *op,
                                             std::shared_ptr<const sh::filesystem::Eurl> eurl)
                                             override

void itemlist (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
               sh::filesystem::FilesystemNodeType type, sh::filesystem::FilesystemNodeListEditor
               *list)

void itemlist (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
               sh::filesystem::FilesystemNodeType type, sh::filesystem::FilesystemNodeListEditor
               *list) = 0
    Determine a list of subelements in a certain directory.

sh::filesystem::FilesystemNodeType getType (sh::filesystem::Operation *op,
                                             std::shared_ptr<const sh::filesystem::Eurl> eurl) =
                                             0
    Determine if a file is regular, or a dir, or a link, ...

qint64 getSize (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl)

qint64 getSize (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl)
        = 0
    Determine the size of a file.

QDateTime getMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl)

QDateTime getMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl) = 0
    Determine the mtime of a file.

void setMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
               QDateTime time)

void setMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
               QDateTime time) = 0
    Set the mtime of a file.

QString getMimeType (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl)

QString getMimeType (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) = 0
    Determine mime type of a file.

QString getLinkTarget (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl)

QString getLinkTarget (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) = 0
    Resolve a link.

bool canGetFileContent (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl)

bool canGetFileContent (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) = 0
    Returns if it is allowed to get the content of a certain file.

std::shared_ptr<QIODevice> getFileContent (sh::filesystem::Operation *op,
                                           std::shared_ptr<const sh::filesystem::Eurl>
                                           eurl)
```



```

std::shared_ptr<QIODevice> getFileContent (sh::filesystem::Operation *op,
                                           std::shared_ptr<const sh::filesystem::Eurl>
                                           eurl) = 0
    Get the content of a file.

bool canCreateDirectory (sh::filesystem::Operation *op, std::shared_ptr<const
                        sh::filesystem::Eurl> eurl)

bool canCreateDirectory (sh::filesystem::Operation *op, std::shared_ptr<const
                        sh::filesystem::Eurl> eurl) = 0
    Returns if it is allowed to create subdirectories in a certain directory.

void createDirectory (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl)

void createDirectory (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) = 0
    Create a directory.

bool canCreateLink (sh::filesystem::Operation *op, std::shared_ptr<const
                   sh::filesystem::Eurl> eurl)

bool canCreateLink (sh::filesystem::Operation *op, std::shared_ptr<const
                   sh::filesystem::Eurl> eurl) = 0
    Returns if it is allowed to create a link in a certain directory.

void createLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                eurl, QString target)

void createLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                eurl, QString target) = 0
    Create a link.

bool canCreateFile (sh::filesystem::Operation *op, std::shared_ptr<const
                   sh::filesystem::Eurl> eurl)

bool canCreateFile (sh::filesystem::Operation *op, std::shared_ptr<const
                   sh::filesystem::Eurl> eurl) = 0
    Returns if it is allowed to create files in a certain directory.

void createFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                eurl, QIODevice *content, HandlerTransfer *handlertransfer)

void createFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                eurl, QIODevice *content, HandlerTransfer *handlertransfer) = 0
    Creates a file with some content.

    It may use handlertransfer for some better integration, like cancel support and progress monitoring.

bool canDeleteItem (sh::filesystem::Operation *op, std::shared_ptr<const
                   sh::filesystem::Eurl> eurl)

bool canDeleteItem (sh::filesystem::Operation *op, std::shared_ptr<const
                   sh::filesystem::Eurl> eurl) = 0
    Returns if it is allowed to delete a certain file/directory/link/...

void deleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                eurl)

void deleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                eurl) = 0
    Delete a file/directory/link/...

bool canRenameItem (sh::filesystem::Operation *op, std::shared_ptr<const
                   sh::filesystem::Eurl> src)

```

bool **canRenameItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> src) = 0

Returns if it is allowed to rename a certain file, directory, link, ... (see renameItem).

void **renameItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> src, QString destpath)

void **renameItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> src, QString destpath) = 0

Renames an item to another path.

It can be a simple filename change or also changes in the deeper path segments.

QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> **getActions** (const QList<std::shared\_ptr<const *sh::filesystem::Eurl*>> eurls)

QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> **getActions** (const QList<std::shared\_ptr<const *sh::filesystem::Eurl*>> eurls) = 0

Returns a list of actions (see former example) for showing for certain files.

bool **requiresResolvedLinks** ()

Returns if this handler requires to be used by the engine with resolved links.

bool **isWellKnownScheme** ()

Returns if the scheme for this handler is a well known one (which external programs typically would understand).

void **configureItems** (*sh::filesystem::Operation* \*op, QList<std::shared\_ptr<FilesystemNode>> nodes)

Configure newly created nodes (e.g. setting another icon or changing the display name).

QList<QString> **listExtendedAttributes** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)

Fetches the list of available extended attributes for an entry.

quint64 **getExtendedAttributeSize** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QString attribute)

Returns the size of value for one extended attribute for an entry.

QByteArray **getExtendedAttribute** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QString attribute)

Returns the value for one extended attribute for an entry.

void **setExtendedAttribute** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QString attribute, QByteArray value)

Sets the value for one extended attribute for an entry.

void **removeExtendedAttribute** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QString attribute)

Removes one extended attributes for an entry.

QMap<QString, QString> **getCustomAttributes** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)

Returns the custom attributes for an entry.

In contrast to extended attributes, the custom attributes are not directly stored in the filesystem this way, but handle implementation specific aspects (like permissions).



void **setCustomAttribute** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QString attribute, QString value)  
 Sets the value for one custom attribute for an entry.  
 See also getCustomAttributes.

void **deleteDirectoryIfEmpty** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)  
 Deletes a directory only if it is empty.

void **viewEnteredDirectory** (std::shared\_ptr<const *sh::filesystem::Eurl*>)  
 Callback for preparing stuff when the view entered some directory (e.g. for enabling watchers). See also viewLeftDirectory.

void **viewLeftDirectory** (std::shared\_ptr<const *sh::filesystem::Eurl*>)  
 Callback for preparing stuff when the view left some directory. See also viewEnteredDirectory.

void **search** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, std::function<void> std::shared\_ptr<const *sh::filesystem::Eurl*>, QList<std::shared\_ptr<*sh::search::SearchCriterion*>>, *sh::filesystem::FilesystemNodeType* ftype  
 > addfile, std::function<void>std::shared\_ptr<const *sh::filesystem::Eurl*>> visitdir, QList<std::shared\_ptr<*sh::search::SearchCriterion*>> criteria)  
 Helps searching for files.  
 Override this method with a behavior identical to the default, if a specialized implementation can be much faster than the default one.

void **customizeUi** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> node, bool \*showSearch-Panel)  
 Creates a custom gui, which becomes part of the fileview.

*sh::filesystem::FilesystemModel* \***model** ()  
 A convenience shortcut to the *sh::filesystem::FilesystemModel* (a singleton).

void **doShutdown** ()  
 Executes singleton shutdown.

void **shutdown** ()  
 Shutdown down this singleton.

bool **isAlive** ()  
 Returns if this singleton is alive (true until its shutdown begins).

## Public Static Functions

QString **mountDevice** (QString gvolumeuuid, *sh::actions::ActionExecutionInfo* \*info)  
 QString **mountLocation** (QString gurl, *sh::actions::ActionExecutionInfo* \*info)  
 QString **unmountLocation** (QString gurl, *sh::actions::ActionExecutionInfo* \*info)  
 QByteArray **eurl2gurlb** (std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)  
 QString **eurl2gurl** (std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)  
 std::shared\_ptr<const *sh::filesystem::Eurl*> **gurl2eurl** (QString gurl)  
 void **logIfError** (void \*error)  
 void **throwIfError** (void \*error)

## Private Functions

**GnomeIOSmbFilesystemHandler()**

**class LocalFilesystemHandler**: public *sh::filesystem::FilesystemHandler*, public *sh::base::Singleton*  
#include <localfilesystemhandler.h> A filesystem handler for the local filesystem.

Subclassed by *sh::filesystemhandlers::WindowsNetworkFilesystemHandler*

## Public Functions

void **itemlist** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl,  
*sh::filesystem::FilesystemNodeType* type, *sh::filesystem::FilesystemNodeListEditor*  
\*list) **override**

void **configureItems** (*sh::filesystem::Operation* \*op, QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>>  
nodes) **override**  
Configure newly created nodes (e.g. setting another icon or changing the display name).

*sh::filesystem::FilesystemNodeType* **getType** (*sh::filesystem::Operation* \*op,  
std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)  
**override**

qint64 **getSize** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)  
**override**

QDateTime **getMtime** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*>  
eurl) **override**

void **setMtime** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl,  
QDateTime mtime) **override**

QString **getMimetype** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl*> eurl) **override**

QString **getLinkTarget** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl*> eurl) **override**

bool **canGetFileContent** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl*> eurl) **override**

std::shared\_ptr<QIODevice> **getFileContent** (*sh::filesystem::Operation* \*op,  
std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) **override**

bool **canCreateDirectory** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl*> eurl) **override**

void **createDirectory** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl*> eurl) **override**

bool **canCreateLink** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl*> eurl) **override**

void **createLink** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*>  
eurl, QString target) **override**

bool **canCreateFile** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl*> eurl) **override**

void **createFile** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*>  
eurl, QIODevice \*content, HandlerTransfer \*handlertransfer) **override**

bool **canDeleteItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl*> eurl) **override**

```

void deleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                eurl) override

bool canRenameItem (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> src) override

void renameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> src,
                 QString destpath) override

QList<std::shared_ptr<sh::actions::AbstractActionItem>> getActions (const
                                                                QList<std::shared_ptr<const
                                                                sh::filesystem::Eurl>> eurls) override

QList<QString> listExtendedAttributes (sh::filesystem::Operation *op,
                                       std::shared_ptr<const sh::filesystem::Eurl>
                                       eurl) override

quint64 getExtendedAttributeSize (sh::filesystem::Operation *op, std::shared_ptr<const
                                  sh::filesystem::Eurl> eurl, QString attribute)
                                  override

QByteArray getExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
                                  sh::filesystem::Eurl> eurl, QString attribute)
                                  override

void setExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
                           sh::filesystem::Eurl> eurl, QString attribute, QByteArray
                           value) override

void removeExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
                              sh::filesystem::Eurl> eurl, QString attribute) override

QMap<QString, QString> getCustomAttributes (sh::filesystem::Operation
                                              *op, std::shared_ptr<const
                                              sh::filesystem::Eurl> eurl) override

void setCustomAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
                          sh::filesystem::Eurl> eurl, QString attribute, QString value)
                          override

bool requiresResolvedLinks () override
    Returns if this handler requires to be used by the engine with resolved links.

bool isWellKnownScheme () override
    Returns if the scheme for this handler is a well known one (which external programs typically would
    understand).

QString eurlToLocal (std::shared_ptr<const sh::filesystem::Eurl> eurl)

std::shared_ptr<const sh::filesystem::Eurl> localToEurl (const QString local)

void itemlist (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
               sh::filesystem::FilesystemNodeType type, sh::filesystem::FilesystemNodeListEditor
               *list) = 0
    Determine a list of subelements in a certain directory.

sh::filesystem::FilesystemNodeType getType (sh::filesystem::Operation *op,
                                              std::shared_ptr<const sh::filesystem::Eurl> eurl) =
                                              0
    Determine if a file is regular, or a dir, or a link, ...

quint64 getSize (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl)
          = 0
    Determine the size of a file.

```

`QDateTime` **getMtime** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*) = 0  
Determine the mtime of a file.

void **setMtime** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*,  
QDateTime *time*) = 0  
Set the mtime of a file.

`QString` **getMimeType** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl*> *eurl*) = 0  
Determine mime type of a file.

`QString` **getLinkTarget** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl*> *eurl*) = 0  
Resolve a link.

`QList<QString>` **listExtendedAttributes** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl*> *eurl*)  
Fetches the list of available extended attributes for an entry.

quint64 **getExtendedAttributeSize** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl*> *eurl*, `QString` *attribute*)  
Returns the size of value for one extended attribute for an entry.

`QByteArray` **getExtendedAttribute** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl*> *eurl*, `QString` *attribute*)  
Returns the value for one extended attribute for an entry.

void **setExtendedAttribute** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl*> *eurl*, `QString` *attribute*, `QByteArray` *value*)  
Sets the value for one extended attribute for an entry.

void **removeExtendedAttribute** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl*> *eurl*, `QString` *attribute*)  
Removes one extended attributes for an entry.

`QMap<QString, QString>` **getCustomAttributes** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl*> *eurl*)  
Returns the custom attributes for an entry.  
  
In contrast to extended attributes, the custom attributes are not directly stored in the filesystem this way, but handle implementation specific aspects (like permissions).

void **setCustomAttribute** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl*> *eurl*, `QString` *attribute*, `QString` *value*)  
Sets the value for one custom attribute for an entry.  
  
See also `getCustomAttributes`.

bool **canGetFileContent** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl*> *eurl*) = 0  
Returns if it is allowed to get the content of a certain file.

std::shared\_ptr<`QIODevice`> **getFileContent** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl*> *eurl*) = 0  
Get the content of a file.

bool **canCreateDirectory** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
*sh::filesystem::Eurl*> *eurl*) = 0  
Returns if it is allowed to create subdirectories in a certain directory.

```

void createDirectory (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) = 0
    Create a directory.

bool canCreateLink (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) = 0
    Returns if it is allowed to create a link in a certain directory.

void createLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                eurl, QString target) = 0
    Create a link.

bool canCreateFile (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) = 0
    Returns if it is allowed to create files in a certain directory.

void createFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                eurl, QIODevice *content, HandlerTransfer *handlertransfer) = 0
    Creates a file with some content.

    It may use handlertransfer for some better integration, like cancel support and progress moni-
    toring.

bool canDeleteItem (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) = 0
    Returns if it is allowed to delete a certain file/directory/link/...

void deleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                eurl) = 0
    Delete a file/directory/link/...

void deleteDirectoryIfEmpty (sh::filesystem::Operation *op, std::shared_ptr<const
                            sh::filesystem::Eurl> eurl)
    Deletes a directory only if it is empty.

bool canRenameItem (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> src) = 0
    Returns if it is allowed to rename a certain file, directory, link, ... (see renameItem).

void renameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> src,
                QString destpath) = 0
    Renames an item to another path.

    It can be a simple filename change or also changes in the deeper path segments.

QList<std::shared_ptr<sh::actions::AbstractActionItem>> getActions (const
                                                                QList<std::shared_ptr<const
                                                                sh::filesystem::Eurl>> eu-
                                                                rls) = 0
    Returns a list of actions (see former example) for showing for certain files.

void viewEnteredDirectory (std::shared_ptr<const sh::filesystem::Eurl>)
    Callback for preparing stuff when the view entered some directory (e.g. for enabling watchers). See
    also viewLeftDirectory.

void viewLeftDirectory (std::shared_ptr<const sh::filesystem::Eurl>)
    Callback for preparing stuff when the view left some directory. See also viewEnteredDirectory.

void search (sh::filesystem::Operation *op, std::shared_ptr<const
            sh::filesystem::Eurl> eurl, std::function<void> std::shared_ptr<const
            sh::filesystem::Eurl>, QList<std::shared_ptr<sh::search::SearchCriterion>>,
            sh::filesystem::FilesystemNodeType ftype
            > addfile, std::function<void> std::shared_ptr<const sh::filesystem::Eurl>> visitdir,
            QList<std::shared_ptr<sh::search::SearchCriterion>> criteria)
    Helps searching for files.

```

Override this method with a behavior identical to the default, if a specialized implementation can be much faster than the default one.

void **customizeUi** (std::shared\_ptr<sh::filesystem::FileSystemNode> node, bool \*showSearch-Panel)

Creates a custom gui, which becomes part of the fileview.

sh::filesystem::FileSystemModel \***model** ()

A convenience shortcut to the sh::filesystem::FileSystemModel (a singleton).

void **doShutdown** ()

Executes singleton shutdown.

void **shutdown** ()

Shutdown down this singleton.

bool **isAlive** ()

Returns if this singleton is alive (true until its shutdown begins).

**class SharcFileSystemHandler** : public sh::filesystem::FileSystemHandler, public sh::base::Singleton  
#include <sharcfilesystemhandler.h> A filesystem handler for the sharc archives.

This is a brutally easy and inefficient archive format which exists mostly for testing.

## Public Functions

void **itemlist** (sh::filesystem::Operation \*op, std::shared\_ptr<const sh::filesystem::Eurl> eurl, sh::filesystem::FileSystemNodeType type, sh::filesystem::FileSystemNodeListEditor \*list) **override**

void **configureItems** (sh::filesystem::Operation \*op, QList<std::shared\_ptr<sh::filesystem::FileSystemNode>> nodes) **override**

sh::filesystem::FileSystemNodeType **getType** (sh::filesystem::Operation \*op, std::shared\_ptr<const sh::filesystem::Eurl> eurl) **override**

qint64 **getSize** (sh::filesystem::Operation \*op, std::shared\_ptr<const sh::filesystem::Eurl> eurl) **override**

QDateTime **getMtime** (sh::filesystem::Operation \*op, std::shared\_ptr<const sh::filesystem::Eurl> eurl) **override**

void **setMtime** (sh::filesystem::Operation \*op, std::shared\_ptr<const sh::filesystem::Eurl> eurl, QDateTime time) **override**

QString **getMimeType** (sh::filesystem::Operation \*op, std::shared\_ptr<const sh::filesystem::Eurl> eurl) **override**

QString **getLinkTarget** (sh::filesystem::Operation \*op, std::shared\_ptr<const sh::filesystem::Eurl> eurl) **override**

bool **canGetFileContent** (sh::filesystem::Operation \*op, std::shared\_ptr<const sh::filesystem::Eurl> eurl) **override**

std::shared\_ptr<QIODevice> **getFileContent** (sh::filesystem::Operation \*op, std::shared\_ptr<const sh::filesystem::Eurl> eurl) **override**

bool **canCreateDirectory** (sh::filesystem::Operation \*op, std::shared\_ptr<const sh::filesystem::Eurl> eurl) **override**

void **createDirectory** (sh::filesystem::Operation \*op, std::shared\_ptr<const sh::filesystem::Eurl> eurl) **override**



```

bool canCreateLink (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) override

void createLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                eurl, QString target) override

bool canCreateFile (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) override

void createFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                eurl, QIODevice *content, HandlerTransfer *handlertransfer) override

bool canDeleteItem (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) override

void deleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                eurl) override

bool canRenameItem (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> src) override

void renameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> src,
                 QString destpath) override

QList<std::shared_ptr<sh::actions::AbstractActionItem>> getActions (const
                                                                    QList<std::shared_ptr<const
                                                                    sh::filesystem::Eurl>> eu-
                                                                    rls) override

void itemlist (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
               sh::filesystem::FilesystemNodeType type, sh::filesystem::FilesystemNodeListEditor
               *list) = 0
    Determine a list of subelements in a certain directory.

void configureItems (sh::filesystem::Operation *op, QList<std::shared_ptr<FilesystemNode>>
                    nodes)
    Configure newly created nodes (e.g. setting another icon or changing the display name).

sh::filesystem::FilesystemNodeType getType (sh::filesystem::Operation *op,
                                              std::shared_ptr<const sh::filesystem::Eurl> eurl) =
                                              0
    Determine if a file is regular, or a dir, or a link, ...

qint64 getSize (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl)
              = 0
    Determine the size of a file.

QDateTime getMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl) = 0
    Determine the mtime of a file.

void setMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
               QDateTime time) = 0
    Set the mtime of a file.

QString getMimetype (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) = 0
    Determine mime type of a file.

QString getLinkTarget (sh::filesystem::Operation *op, std::shared_ptr<const
                       sh::filesystem::Eurl> eurl) = 0
    Resolve a link.

QList<QString> listExtendedAttributes (sh::filesystem::Operation *op,
                                         std::shared_ptr<const sh::filesystem::Eurl>
                                         eurl)

```

Fetches the list of available extended attributes for an entry.

```
quint64 getExtendedAttributeSize (sh::filesystem::Operation *op, std::shared_ptr<const
                                     sh::filesystem::Eurl> eurl, QString attribute)
```

Returns the size of value for one extended attribute for an entry.

```
QByteArray getExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
                                   sh::filesystem::Eurl> eurl, QString attribute)
```

Returns the value for one extended attribute for an entry.

```
void setExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
                           sh::filesystem::Eurl> eurl, QString attribute, QByteArray
                           value)
```

Sets the value for one extended attribute for an entry.

```
void removeExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
                              sh::filesystem::Eurl> eurl, QString attribute)
```

Removes one extended attributes for an entry.

```
QMap<QString, QString> getCustomAttributes (sh::filesystem::Operation
                                              *op, std::shared_ptr<const
                                              sh::filesystem::Eurl> eurl)
```

Returns the custom attributes for an entry.

In contrast to extended attributes, the custom attributes are not directly stored in the filesystem this way, but handle implementation specific aspects (like permissions).

```
void setCustomAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
                          sh::filesystem::Eurl> eurl, QString attribute, QString value)
```

Sets the value for one custom attribute for an entry.

See also getCustomAttributes.

```
bool canGetFileContent (sh::filesystem::Operation *op, std::shared_ptr<const
                       sh::filesystem::Eurl> eurl) = 0
```

Returns if it is allowed to get the content of a certain file.

```
std::shared_ptr<QIODevice> getFileContent (sh::filesystem::Operation *op,
                                           std::shared_ptr<const sh::filesystem::Eurl>
                                           eurl) = 0
```

Get the content of a file.

```
bool canCreateDirectory (sh::filesystem::Operation *op, std::shared_ptr<const
                          sh::filesystem::Eurl> eurl) = 0
```

Returns if it is allowed to create subdirectories in a certain directory.

```
void createDirectory (sh::filesystem::Operation *op, std::shared_ptr<const
                      sh::filesystem::Eurl> eurl) = 0
```

Create a directory.

```
bool canCreateLink (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) = 0
```

Returns if it is allowed to create a link in a certain directory.

```
void createLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                 eurl, QString target) = 0
```

Create a link.

```
bool canCreateFile (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) = 0
```

Returns if it is allowed to create files in a certain directory.

```
void createFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                 eurl, QIODevice *content, HandlerTransfer *handlertransfer) = 0
```

Creates a file with some content.



It may use `handlertransfer` for some better integration, like cancel support and progress monitoring.

```
bool canDeleteItem (sh::filesystem::Operation *op, std::shared_ptr<const
sh::filesystem::Eurl> eurl) = 0
```

Returns if it is allowed to delete a certain file/directory/link/...

```
void deleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
eurl) = 0
```

Delete a file/directory/link/...

```
void deleteDirectoryIfEmpty (sh::filesystem::Operation *op, std::shared_ptr<const
sh::filesystem::Eurl> eurl)
```

Deletes a directory only if it is empty.

```
bool canRenameItem (sh::filesystem::Operation *op, std::shared_ptr<const
sh::filesystem::Eurl> src) = 0
```

Returns if it is allowed to rename a certain file, directory, link, ... (see `renameItem`).

```
void renameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> src,
QString destpath) = 0
```

Renames an item to another path.

It can be a simple filename change or also changes in the deeper path segments.

```
QList<std::shared_ptr<sh::actions::AbstractActionItem>> getActions (const
                                                                    QList<std::shared_ptr<const
                                                                    sh::filesystem::Eurl>> eu-
                                                                    rls) = 0
```

Returns a list of actions (see former example) for showing for certain files.

```
bool requiresResolvedLinks ()
```

Returns if this handler requires to be used by the engine with resolved links.

```
void viewEnteredDirectory (std::shared_ptr<const sh::filesystem::Eurl>)
```

Callback for preparing stuff when the view entered some directory (e.g. for enabling watchers). See also `viewLeftDirectory`.

```
void viewLeftDirectory (std::shared_ptr<const sh::filesystem::Eurl>)
```

Callback for preparing stuff when the view left some directory. See also `viewEnteredDirectory`.

```
bool isWellKnownScheme ()
```

Returns if the scheme for this handler is a well known one (which external programs typically would understand).

```
void search (sh::filesystem::Operation *op, std::shared_ptr<const
sh::filesystem::Eurl> eurl, std::function<void> std::shared_ptr<const
sh::filesystem::Eurl>, QList<std::shared_ptr<sh::search::SearchCriterion>>,
sh::filesystem::FilesystemNodeType ftype
> addfile, std::function<void>std::shared_ptr<const sh::filesystem::Eurl>> visitdir,
QList<std::shared_ptr<sh::search::SearchCriterion>> criteria
```

Helps searching for files.

Override this method with a behavior identical to the default, if a specialized implementation can be much faster than the default one.

```
void customizeUi (std::shared_ptr<sh::filesystem::FilesystemNode> node, bool *showSearch-
Panel)
```

Creates a custom gui, which becomes part of the fileview.

```
sh::filesystem::FilesystemModel *model ()
```

A convenience shortcut to the `sh::filesystem::FilesystemModel` (a singleton).

```
void doShutdown ()
```

Executes singleton shutdown.

void **shutdown** ()  
    Shutdown down this singleton.

bool **isAlive** ()  
    Returns if this singleton is alive (true until its shutdown begins).

## Public Static Functions

std::shared\_ptr<const *filesystem::Eurl*> **localToEurl** (const QString *local*)

## Private Functions

**SharcFilesystemHandler** ()

**class WindowsNetworkFilesystemHandler** : public *sh::filesystemhandlers::LocalFilesystemHandler*  
    #include <windowsnetworkfilesystemhandler.h>

## Public Functions

void **itemlist** (*sh::filesystem::Operation* \**op*, std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*,  
                *sh::filesystem::FilesystemNodeType* *type*, *sh::filesystem::FilesystemNodeListEditor*  
                \**list*) **override**

void **itemlist** (*sh::filesystem::Operation* \**op*, std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*,  
                *sh::filesystem::FilesystemNodeType* *type*, *sh::filesystem::FilesystemNodeListEditor*  
                \**list*) = 0  
    Determine a list of subelements in a certain directory.

void **configureItems** (*sh::filesystem::Operation* \**op*, QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>>  
                      *nodes*) **override**  
    Configure newly created nodes (e.g. setting another icon or changing the display name).

*sh::filesystem::FilesystemNodeType* **getType** (*sh::filesystem::Operation* \**op*,  
  std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*)  
  **override**

*sh::filesystem::FilesystemNodeType* **getType** (*sh::filesystem::Operation* \**op*,  
  std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*) =  
  0  
    Determine if a file is regular, or a dir, or a link, ...

qint64 **getSize** (*sh::filesystem::Operation* \**op*, std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*)  
                    **override**

qint64 **getSize** (*sh::filesystem::Operation* \**op*, std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*)  
                    = 0  
    Determine the size of a file.

QDateTime **getMtime** (*sh::filesystem::Operation* \**op*, std::shared\_ptr<const *sh::filesystem::Eurl*>  
                      *eurl*) **override**

QDateTime **getMtime** (*sh::filesystem::Operation* \**op*, std::shared\_ptr<const *sh::filesystem::Eurl*>  
                      *eurl*) = 0  
    Determine the mtime of a file.

void **setMtime** (*sh::filesystem::Operation* \**op*, std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*,  
                QDateTime *mtime*) **override**

void **setMtime** (*sh::filesystem::Operation* \**op*, std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*,  
                QDateTime *time*) = 0  
    Set the mtime of a file.

```

QString getMimeType (sh::filesystem::Operation *op,          std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) override
QString getMimeType (sh::filesystem::Operation *op,          std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) = 0
    Determine mime type of a file.

QString getLinkTarget (sh::filesystem::Operation *op,          std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) override
QString getLinkTarget (sh::filesystem::Operation *op,          std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) = 0
    Resolve a link.

bool canGetFileContent (sh::filesystem::Operation *op,          std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) override
bool canGetFileContent (sh::filesystem::Operation *op,          std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) = 0
    Returns if it is allowed to get the content of a certain file.

std::shared_ptr<QIODevice> getFileContent (sh::filesystem::Operation *op,
                                           std::shared_ptr<const sh::filesystem::Eurl>
                                           eurl) override
std::shared_ptr<QIODevice> getFileContent (sh::filesystem::Operation *op,
                                           std::shared_ptr<const sh::filesystem::Eurl>
                                           eurl) = 0
    Get the content of a file.

bool canCreateDirectory (sh::filesystem::Operation *op,          std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) override
bool canCreateDirectory (sh::filesystem::Operation *op,          std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) = 0
    Returns if it is allowed to create subdirectories in a certain directory.

void createDirectory (sh::filesystem::Operation *op,          std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) override
void createDirectory (sh::filesystem::Operation *op,          std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) = 0
    Create a directory.

bool canCreateLink (sh::filesystem::Operation *op,          std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) override
bool canCreateLink (sh::filesystem::Operation *op,          std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) = 0
    Returns if it is allowed to create a link in a certain directory.

void createLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                 eurl, QString target) override
void createLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                 eurl, QString target) = 0
    Create a link.

bool canCreateFile (sh::filesystem::Operation *op,          std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) override
bool canCreateFile (sh::filesystem::Operation *op,          std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) = 0
    Returns if it is allowed to create files in a certain directory.

void createFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                 eurl, QIODevice *content, HandlerTransfer *handlertransfer) override

```

```
void createFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                eurl, QIODevice *content, HandlerTransfer *handlertransfer) = 0
```

Creates a file with some content.

It may use handlertransfer for some better integration, like cancel support and progress monitoring.

```
bool canDeleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl) override
```

```
bool canDeleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl) = 0
```

Returns if it is allowed to delete a certain file/directory/link/...

```
void deleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl) override
```

```
void deleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl) = 0
```

Delete a file/directory/link/...

```
bool canRenameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> src) override
```

```
bool canRenameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> src) = 0
```

Returns if it is allowed to rename a certain file, directory, link, ... (see renameItem).

```
void renameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> src,
                QString destpath) override
```

```
void renameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> src,
                QString destpath) = 0
```

Renames an item to another path.

It can be a simple filename change or also changes in the deeper path segments.

```
QList<std::shared_ptr<sh::actions::AbstractActionItem>> getActions (const
                                                                QList<std::shared_ptr<const sh::filesystem::Eurl>> eurls) override
```

```
QList<std::shared_ptr<sh::actions::AbstractActionItem>> getActions (const
                                                                QList<std::shared_ptr<const sh::filesystem::Eurl>> eurls) = 0
```

Returns a list of actions (see former example) for showing for certain files.

```
QList<QString> listExtendedAttributes (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl) override
```

```
QList<QString> listExtendedAttributes (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl)
```

Fetches the list of available extended attributes for an entry.

```
quint64 getExtendedAttributeSize (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
                                   QString attribute) override
```

```
quint64 getExtendedAttributeSize (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
                                   QString attribute)
```

Returns the size of value for one extended attribute for an entry.

```
QByteArray getExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
                                sh::filesystem::Eurl> eurl, QString attribute)
                                override
```

```
QByteArray getExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
                                sh::filesystem::Eurl> eurl, QString attribute)
```

Returns the value for one extended attribute for an entry.

```
void setExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
                           sh::filesystem::Eurl> eurl, QString attribute, QByteArray
                           value) override
```

```
void setExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
                           sh::filesystem::Eurl> eurl, QString attribute, QByteArray
                           value)
```

Sets the value for one extended attribute for an entry.

```
void removeExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
                              sh::filesystem::Eurl> eurl, QString attribute) override
```

```
void removeExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
                              sh::filesystem::Eurl> eurl, QString attribute)
```

Removes one extended attributes for an entry.

```
QMap<QString, QString> getCustomAttributes (sh::filesystem::Operation
                                              *op, std::shared_ptr<const
                                              sh::filesystem::Eurl> eurl) override
```

```
QMap<QString, QString> getCustomAttributes (sh::filesystem::Operation
                                              *op, std::shared_ptr<const
                                              sh::filesystem::Eurl> eurl)
```

Returns the custom attributes for an entry.

In contrast to extended attributes, the custom attributes are not directly stored in the filesystem this way, but handle implementation specific aspects (like permissions).

```
void setCustomAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
                          sh::filesystem::Eurl> eurl, QString attribute, QString value)
                          override
```

```
void setCustomAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
                          sh::filesystem::Eurl> eurl, QString attribute, QString value)
```

Sets the value for one custom attribute for an entry.

See also getCustomAttributes.

```
bool requiresResolvedLinks () override
```

Returns if this handler requires to be used by the engine with resolved links.

```
bool isWellKnownScheme () override
```

Returns if the scheme for this handler is a well known one (which external programs typically would understand).

```
QString eurlToLocal (std::shared_ptr<const sh::filesystem::Eurl> eurl)
```

```
std::shared_ptr<const sh::filesystem::Eurl> localToEurl (const QString local)
```

```
void deleteDirectoryIfEmpty (sh::filesystem::Operation *op, std::shared_ptr<const
                              sh::filesystem::Eurl> eurl)
```

Deletes a directory only if it is empty.

```
void viewEnteredDirectory (std::shared_ptr<const sh::filesystem::Eurl>)
```

Callback for preparing stuff when the view entered some directory (e.g. for enabling watchers). See also viewLeftDirectory.

void **viewLeftDirectory** (std::shared\_ptr<const *sh::filesystem::Eurl*>)  
    Callback for preparing stuff when the view left some directory. See also viewEnteredDirectory.

void **search** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const  
    *sh::filesystem::Eurl*> eurl, std::function<void> std::shared\_ptr<const  
    *sh::filesystem::Eurl*>, QList<std::shared\_ptr<*sh::search::SearchCriterion*>>,  
    *sh::filesystem::FilesystemNodeType* ftype  
> addfile, std::function<void>std::shared\_ptr<const *sh::filesystem::Eurl*>> visitdir,  
    QList<std::shared\_ptr<*sh::search::SearchCriterion*>> criteriaHelps searching for files.

Override this method with a behavior identical to the default, if a specialized implementation can be much faster than the default one.

void **customizeUi** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> node, bool \*showSearch-  
    Panel)  
    Creates a custom gui, which becomes part of the fileview.

*sh::filesystem::FilesystemModel* \***model** ()  
    A convenience shortcut to the *sh::filesystem::FilesystemModel* (a singleton).

void **doShutdown** ()  
    Executes singleton shutdown.

void **shutdown** ()  
    Shutdown down this singleton.

bool **isAlive** ()  
    Returns if this singleton is alive (true until its shutdown begins).

### 10.2.13 Namespace *sh::paneldetails*

**namespace** *sh::paneldetails*

Infrastructure for panel details.

Can be shown in the bottom part of the main window. Beyond the infrastructure, it also includes some very basic detail providers.

See *sh::paneldetails::PanelDetailManager* for more.

#### Enums

**enum** **PanelDetailPositionIndex**

Base values for position indexes of panel details.

*Values:*

**enumerator** **VeryInteresting** = 1000000

**enumerator** **Interesting** = 2000000

**enumerator** **Exotic** = 3000000

**class** **CommonPanelDetails**

#include <commonpaneldetails.h> Implementations of common details provider for the details panel.

## Public Static Functions

```

void doInitialize ()
void doShutdown ()

void _ViaDetailColumn (std::shared_ptr<PanelDetail> detail,
                      std::shared_ptr<sh::filesystem::FilesystemNode> node,
                      std::shared_ptr<sh::filesystem::DetailColumn> column, QString
                      keyname, std::function<QString> QVariant
                      > valtostring)

void NumberOfSelectedItems (std::shared_ptr<PanelDetail>, QList<std::shared_ptr<sh::filesystem::FilesystemNode>,
                          sh::filesystem::Operation *op)

void TotalFileSize (std::shared_ptr<PanelDetail>, QList<std::shared_ptr<sh::filesystem::FilesystemNode>>,
                    sh::filesystem::Operation *op)

void FileType (std::shared_ptr<PanelDetail> detail, std::shared_ptr<sh::filesystem::FilesystemNode>
               node, sh::filesystem::Operation *op)

void FileSize (std::shared_ptr<PanelDetail> detail, std::shared_ptr<sh::filesystem::FilesystemNode>
               node, sh::filesystem::Operation *op)

void FileModificationTime (std::shared_ptr<PanelDetail> detail,
                           std::shared_ptr<sh::filesystem::FilesystemNode> node,
                           sh::filesystem::Operation *op)

class PanelDetail : public QObject
    #include <paneldetailmanager.h> A detail panel entry.

    Is is essentially a list of PanelDetailRow and some layout information.

```

## Public Functions

```

PanelDetail (int positionIndex, int valueWidthHint)
    Constructed only by the infrastructure and made available otherwise.

QList<PanelDetailRow> rows ()
    Returns the list of detail rows stored in this panel detail.

int positionIndex ()
    Returns a position index.

    It controls the position of this panel detail in relation to the other ones.

int valueWidthHint ()
    Returns the specified width for the panel detail.

void setRows (QList<PanelDetailRow> rows)
    Sets the list of detail rows.

void addRow (PanelDetailRow row)
    Adds a new detail row.

void addOrReplaceRow (PanelDetailRow row)
    Adds a new detail row (replacing an old one with the same key).

void _emit_linkTriggered (QString link)

```



## Signals

void **changed** ()  
Emitted when data has changed.

void **linkTriggered** (QString *link*)  
Emitted when a link is triggered by the user.

## Private Members

QMutex **\_rowsmutex**

QList<PanelDetailRow> **\_rows**

int **\_positionIndex**

int **\_valueWidthHint**

**class PanelDetailFactory**  
*#include <paneldetailmanager.h>* Abstract factory for creating panel details for the selected nodes.

Subclassed by *sh::paneldetails::PanelDetailFactoryForFunctionMulti*,  
*sh::paneldetails::PanelDetailFactoryForFunctionSingle*

## Public Functions

std::shared\_ptr<PanelDetail> **construct** (QList<std::shared\_ptr<sh::filesystem::FilesystemNode>>  
*nodes*, *sh::filesystem::Operation \*op*) = 0

**class PanelDetailFactoryForFunctionMulti : public sh::paneldetails::PanelDetailFactory**  
*#include <paneldetailmanager.h>* A factory for creating panel details for more than one selected node.

Subclassed by *sh::scripting::api::ApiPanelDetailFactoryMulti*

## Public Functions

**PanelDetailFactoryForFunctionMulti** (std::function<void> std::shared\_ptr<PanelDetail>,  
QList<std::shared\_ptr<sh::filesystem::FilesystemNode>>,  
*sh::filesystem::Operation \*op*  
> *fcnsetvalue*, std::function<void> QList<std::shared\_ptr<sh::filesystem::FilesystemNode>>, QString  
*link*> *fcnlinktriggered*, int *position*, int *valueWidthHint*) Constructed only indirectly.

std::shared\_ptr<PanelDetail> **construct** (QList<std::shared\_ptr<sh::filesystem::FilesystemNode>>  
*nodes*, *sh::filesystem::Operation \*op*)

**class PanelDetailFactoryForFunctionSingle : public sh::paneldetails::PanelDetailFactory**  
*#include <paneldetailmanager.h>* A factory for creating panel details for just one selected node.

Subclassed by *sh::scripting::api::ApiPanelDetailFactorySingle*



## Public Functions

```

PanelDetailFactoryForFunctionSingle (std::function<void> std::shared_ptr<PanelDetail>,
                                     std::shared_ptr<sh::filesystem::FilesystemNode>,
                                     sh::filesystem::Operation *op
                                     > fctsetvalue, std::function<void>std::shared_ptr<sh::filesystem::FilesystemNode>, QString link>
                                     fctlinktriggered, int position, int valueWidthHintConstructed only indirectly.

std::shared_ptr<PanelDetail> construct (QList<std::shared_ptr<sh::filesystem::FilesystemNode>>
                                     nodes, sh::filesystem::Operation *op)

class PanelDetailManager : public QObject, public sh::base::Singleton
    #include <paneldetailmanager.h> Manager for details provider as used in the details panel.

    Register some code and data here for implementing a new provider.

```

## Public Functions

```

QList<std::shared_ptr<PanelDetail>> getDetails (QList<std::shared_ptr<sh::filesystem::FilesystemNode>>
                                             nodes)
    Returns a list of PanelDetail for a list of selected filesystem nodes.

std::shared_ptr<PanelDetailFactory> registerFactorySingle (std::shared_ptr<PanelDetailFactory>
                                                         factory)
    Registers a panel detail factory for single selection by PanelDetailFactory instance.

std::shared_ptr<PanelDetailFactory> registerFactoryMulti (std::shared_ptr<PanelDetailFactory>
                                                         factory)
    Registers a panel detail factory for multi selection by PanelDetailFactory instance.

std::shared_ptr<PanelDetailFactory> registerFactorySingle (std::function<void> std::shared_ptr<PanelDetail>,
                                                         std::shared_ptr<sh::filesystem::FilesystemNode>,
                                                         sh::filesystem::Operation *
                                                         > fctsetvalue, std::function<void>std::shared_ptr<sh::filesystem::FilesystemNode>, QString link>
                                                         fctlinktriggered, int position, int valueWidthHintRegisters a panel detail factory for single selection
                                                         by parameters.

std::shared_ptr<PanelDetailFactory> registerFactoryMulti (std::function<void> std::shared_ptr<PanelDetail>,
                                                         QList<std::shared_ptr<sh::filesystem::FilesystemNode>>,
                                                         sh::filesystem::Operation *
                                                         > fctsetvalue, std::function<void>QList<std::shared_ptr<sh::filesystem::FilesystemNode>>, QString
                                                         link> fctlinktriggered, int position, int valueWidthHintRegisters a panel detail factory for multi se-
                                                         lection by parameters.

void doInitialize ()
    Executes singleton initialization.

void doShutdown ()
    Executes singleton shutdown.

void shutdown ()
    Shutdown down this singleton.

bool isAlive ()
    Returns if this singleton is alive (true until its shutdown begins).

```

### Private Functions

**PanelDetailManager** ( )

### Private Members

QList<std::shared\_ptr<*PanelDetailFactory*>> **\_factoryiessingle**

QList<std::shared\_ptr<*PanelDetailFactory*>> **\_factoriesmulti**

QMutex **\_mutex**

**class PanelDetailRow**

*#include <paneldetailmanager.h>* One pair of label text and value text in a panel detail.

### Public Functions

**PanelDetailRow** (QString *key*, QList<*PanelDetailRowValueElement*> *value*)

Is intended to be directly constructed from everywhere.

**PanelDetailRow** (QList<*PanelDetailRowValueElement*> *value*)

Is intended to be directly constructed from everywhere.

QString **key** ( )

QList<*PanelDetailRowValueElement*> **value** ( )

### Private Members

QString **\_key**

QList<*PanelDetailRowValueElement*> **\_value**

**class PanelDetailRowValueElement**

*#include <paneldetailmanager.h>* One element of a detail row's value.

This can be something like a string, an icon or a link button.

### Public Functions

**PanelDetailRowValueElement** ( )

A placeholder for a value which is not yet arrived. It could be visualized with a loading animation or '...'. Is intended to be directly constructed from everywhere.

**PanelDetailRowValueElement** (QString *text*)

A piece of text. Is intended to be directly constructed from everywhere.

**PanelDetailRowValueElement** (QIcon *icon*)

An icon. Is intended to be directly constructed from everywhere.

**PanelDetailRowValueElement** (QString *text*, QString *linktarget*, bool *autohide*)

A link button. Is intended to be directly constructed from everywhere.

QString **text** ( )

Returns the text.

QIcon **icon** ( )

Returns the icon.

QString **linktarget** ()

Returns the link target.

bool **linkautohide** ()

Returns if this link shall automatically hide (e.g. when the mouse disappears).

bool **isWaiting** ()

Returns if it is currently waiting for an update.

### Private Members

QString **\_text**

QIcon **\_icon**

QString **\_linktarget**

bool **\_linkautohide** = false

bool **\_iswaiting** = false

## 10.2.14 Namespace sh::scripting

namespace *sh::scripting*

The Shallot scripting interface.

### Typedefs

using **StringMap** = QMap<QString, V>

class **Api**

*#include <api.h>* Interoperation layer between the Shallot core and a plugin interpreter.

### Public Functions

**Api** ()

**~Api** ()

QStringList **classes** ()

QStringList **rootMembers** ()

*ApiClass* \***getClass** (QString *name*)

*ApiClass* \***getClassByIdName** (QString *typeidName*)

void \***rootMember** (QString *name*)

*ApiClass* \***rootMemberClass** (QString *name*)

### Private Functions

```
void registerClass (QString name, ApiClass **apiclass, const std::type_info &nativeType)  
void registerMethod (ApiClass *apiclass, QString name)  
void registerRootObjectMember (QString name, ApiClass *cls, void *member)
```

### Private Members

```
QHash<QString, ApiClass*> _classes  
QHash<QString, ApiClass*> _classesByNativeType  
QHash<QString, void*> _rootMembers  
QHash<QString, ApiClass*> _rootMembersClasses  
class ApiClass  
    #include <api.h> Data structure for one api-aware core class.
```

### Public Functions

```
ApiClass (QString name)  
    Constructed only indirectly.  
~ApiClass ()  
QString name ()  
QStringList methods ()  
void addMethod (ApiMethod *m)  
ApiMethod *getMethod (QString name)
```

### Private Members

```
QString _name  
QHash<QString, ApiMethod*> _methods  
class ApiMethod  
    #include <api.h> Data structure for one api-aware core method.
```

### Public Functions

```
ApiMethod (QString name)  
    Constructed only indirectly.  
QString name ()  
class PythonScriptInterpreter : public sh::scripting::ScriptInterpreter, public sh::base::Singleton  
    #include <pythonscriptinterpreter.h> Integration of the Python language for scripting.  
    Find also the scripting manual for all details about the scripting interface.
```

## Public Functions

`~PythonScriptInterpreter()`  
`bool isAvailable()`  
`void initializeInterpreter()`  
`QString filesuffix()`  
`void execute (QString script)`  
`void doInitialize()`  
    Executes singleton initialization.  
`void shutdown()`  
    Shutdown down this singleton.  
`bool isAlive()`  
    Returns if this singleton is alive (true until its shutdown begins).

## Public Members

`PyObject * _Exception`  
`PyObject * _CancelException`  
`PyObject * _ThreadAbortException`

## Public Static Functions

`void registerMethod (sh::scripting::ApiClass *clss, QString name, PyCFunction fctptr)`  
`void throwNativeExceptionFromPythonTraceback()`

## Public Static Attributes

`QHash<void*, PyObject*> _buddies`  
`QHash<void*, std::weak_ptr<void>> _natives`

## Private Functions

`PythonScriptInterpreter()`

## Private Static Functions

`PyObject *getApiModuleDef()`

### Private Static Attributes

```
void *apiModuleDef
sh::scripting::Api *api = new sh::scripting::Api()
QHash<QString, PyCFunction> _methods
PyObject *modtraceback
class ScriptingEngine : public QObject, public sh::base::Singleton
#include <scriptingengine.h> The Scripting engine loads scripts by means of the registered interpreters.
```

### Public Functions

```
void loadScript (QString script)
    Loads a plugin script from file.
void doShutdown ()
    Executes singleton shutdown.
void shutdown ()
    Shutdown down this singleton.
bool isAlive ()
    Returns if this singleton is alive (true until its shutdown begins).
```

### Private Functions

```
ScriptingEngine ()
void registerInterpreter (std::shared_ptr<sh::scripting::ScriptInterpreter> interpreter)
```

### Private Members

```
QMap<QString, std::shared_ptr<sh::scripting::ScriptInterpreter>> _interpreters
```

### Private Static Attributes

```
std::shared_ptr<sh::configuration::ConfigurationValue> cfgvalLastStartupFailed = sh::configuration::ConfigurationValue::get()
class ActionPluginLoadCrashedAgain : public sh::actions::ActionActionItem
```

### Public Functions

```
ActionPluginLoadCrashedAgain (QString path, QMutex *mutex)
void execute ()
    Executes this action.
void execute (sh::actions::ActionExecutionInfo *info)
    Executes this action.
QKeySequence shortcutHint ()
    Returns the keyboard shortcut for triggering this action.
```

bool **shortcutHintTriggersOnCurrentDirectory** ()  
 Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcutHint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
 Sets the keyboard shortcut for triggering this action.

QString **text** ()  
 Returns the displayed text for this action.

QString **icon** ()  
 Returns the icon for this action.

bool **enabled** ()  
 Checks if this action is enabled.

bool **isChecked** ()  
 Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
 Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
 Returns the precedence for becoming the default action of a parent submenu.  
 This e.g. leads to a bold label.  
 The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
 Sets the displayed text.

void **setIcon** (QString *icon*)  
 Sets the icon.

void **setEnabled** (bool *enabled*)  
 Sets if the item is enabled.

void **setChecked** (bool *checked*)  
 Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
 Sets the visibility of this item.

bool **visible** ()  
 Checks the visibility of this item (non-recursively).

std::weak\_ptr<AbstractActionItem> **parentAction** ()  
 Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<AbstractActionItem> *parent*)  
 Sets the parent action. .

void **initializeAsync** (std::function<void>  
 > *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
 Synchronously initializes the action.

bool **isInitialized** ()  
 Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

**class ScriptedException**: public *sh::exceptions::Exception*  
*#include <scriptingengine.h>* Exception for errors within the scripting engine and interpreters.

## Public Functions

**ScriptedException** (*sh::exceptions::ExceptionData* data)

QString **message** () const

QString **name** () const

QString **classes** () const

QString **details** () const

QString **callstack** () const

QString **auxiliary** () const

QString **customValue** (QString key) const

bool **isRuntimeException** () const

bool **isProgramException** () const

bool **isRetryable** () const

bool **isResumeable** () const

bool **isDetailsAreInteresting** () const

int **autoRetryRecommendedIn** () const

void **setResumeable** (bool v)

void **setReTryable** (bool v)

void **setCustomValue** (QString key, QString value)

bool **isClass** (QString classname)

*sh::exceptions::ExceptionData* **data** ()

## Public Static Functions

template<class **Handler**>  
std::shared\_ptr<RegisterHandler<*Handler*>> **createRegisterHandler** (*Handler* han-  
dler, QS-  
tack<*Handler*>  
\*stack)

void **executeGuarded** (std::function<void>  
> *fcn* int flags = 0) Executes some code with some standard exection handling around.

### Parameters



- `fcn`: The code to execute guarded.
- `flags`: Flags of *ExecuteGuardFlag* for choosing the behavior.

```
void executeGuarded_errorpanel (std::function<void>
    > fcn, int flags = 0)
```

```
QString _demangle (QString l)
```

```
void exceptionDialog (sh::exceptions::Exception &e, bool *doRetry)
```

### Public Static Attributes

```
const QString UNDEFINED_ERROR_OCCURRED = QObject::tr("An unspecified error occurred.")
```

```
const QString SHALLOT_MUST_CLOSE_TEXT = QObject::tr("Your Shallot process is disturbed by an error and ne
```

```
const QString Value_isShallotException = "_isShallotException"
```

```
class ScriptInterpreter
```

```
#include <scriptingengine.h> The base class for integration of a script language.
```

```
Subclassed by sh::scripting::PythonScriptInterpreter
```

### Public Functions

```
bool isAvailable ()
```

```
void initializeInterpreter ()
```

```
QString filesuffix ()
```

```
void execute (QString)
```

```
~ScriptInterpreter ()
```

```
struct shallot_ShallotObject
```

```
A wrapper around a native object for the Python world.
```

### Public Members

```
PyObject_HEAD void * nativeObject
```

```
std::shared_ptr<void> sharedNativeObject
```

```
bool isReallyShared
```

```
namespace api
```

```
Adapter functions and classes for interaction with scripting.
```

Some programming interfaces in the core are not directly usable for the scripting api. Scripting-friendly surrogates for them are here.

```
class ApiActionActionItem: public sh::actions::ActionActionItem
```

```
#include <apiaction.h>
```

## Public Functions

**ApiActionActionItem** (QString *text*, bool *enabled*, QString *icon*, int *defaultActionPrecedence*)

void **\_\_execute** ()

void **\_\_initializeSync** ()

void **action** (sh::actions::ActionExecutionInfo \**info*) **override**

The action implementation, i.e. what the actions should actually do.

void **initialize** () **override**

Initialize the action. This should make the time-consuming parts, e.g. for determining a label or enabled state.

void **execute** ()

Executes this action.

void **execute** (sh::actions::ActionExecutionInfo \**info*)

Executes this action.

QKeySequence **shortcuthint** ()

Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()

Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)

Sets the keyboard shortcut for triggering this action.

QString **text** ()

Returns the displayed text for this action.

QString **icon** ()

Returns the icon for this action.

bool **enabled** ()

Checks if this action is enabled.

bool **isChecked** ()

Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

```

void setChecked (bool checked)
    Sets if the item is checked (has a cross in the ui).

void setVisible (bool visible)
    Sets the visibility of this item.

bool visible ()
    Checks the visibility of this item (non-recursively).

std::weak_ptr<AbstractActionItem> parentAction ()
    Returns the parent action, if it is added to a container.

void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.

```

## Public Members

```

std::function<void ()> _initialize
std::function<void (sh::actions::ActionExecutionInfo*)> _action

```

## Signals

```

void changed ()
    Emits when some data changed.

```

```

class ApiActionFactory : public sh::actions::AbstractActionFactory
    #include <apipredicatedactionfactory.h>

```

## Public Functions

```

ApiActionFactory (QString category, QList<std::shared_ptr<actions::Predicate>> predi-
    cates)
std::shared_ptr<sh::actions::AbstractActionItem> construct (QList<std::shared_ptr<sh::filesystem::FilesystemNode>
    nodes)
std::shared_ptr<ActionInstantiation> actionAvailable (ActionInstantiation *instantia-
    tion)

```

## Public Members

```
std::function< std::shared_ptr< sh::actions::AbstractActionItem >QList< std::sh  
class ApiDetailColumn: public sh::filesystem::DetailColumn  
#include <apidetailcolumn.h>
```

## Public Functions

**ApiDetailColumn** (QString *name*, QString *displayName*, int *displayIndex*, bool *sort\_doTypediff*, int *defaultWidth*, bool *isRightAligned*)

QVariant **determineValue** (std::shared\_ptr<sh::filesystem::FilesystemNode> *node*,  
sh::filesystem::Operation \**op*)

void **applyValue** (std::shared\_ptr<const sh::filesystem::Eurl> *eurl*,  
sh::filesystem::Operation \**op*, QVariant *value*)

QString **name** ()  
The internal unique name.

QString **displayName** ()  
The display name.

bool **isValueAvailable** (std::shared\_ptr<FilesystemNode> *node*)  
Checks if a value for this detail is available for one given node.

QVariant **value** (std::shared\_ptr<FilesystemNode> *node*, bool *ignoreAged* = false)  
Returns the value for this detail for one given node.

### Parameters

- *ignoreAged*: If no value should be returned which is older than the last request (not useful for nearly all cases).

QString **displayValue** (std::shared\_ptr<FilesystemNode> *node*, const  
sh::filesystem::FilesystemModelFileviewProxy \**viewmodel*)  
Returns the stringified value for this details for one given node considering the configuration of a view.

bool **isVisible** ()  
If this detail shall be a visible column in the view.

QVariant **requestValue** (std::shared\_ptr<FilesystemNode> *node*, sh::filesystem::Operation  
\**op* = 0, bool *withNodeValues* = false, bool *withOperationsCache*  
= true)  
Requests to determine the value for this detail for one given node. Blocks until the value is available. Get it with the value method afterwards.

bool **sort\_doTypediff** ()  
If sorting should separate files and directories.

int **sort\_order** (std::shared\_ptr<FilesystemNode> *n1*, std::shared\_ptr<FilesystemNode> *n2*)  
The sorting order.

uint **displayIndex** ()  
Controls which place this column should get in the user interface.

QVariant **computeValue** (std::shared\_ptr<FilesystemNode> *node*, sh::filesystem::Operation  
\**op*)  
Returns a freshly determined value for this column and the given node. It neither asks nor populates any caches or storages.

```
int defaultWidth ()
```

```
bool isRightAligned ()
```

```
void applyValueByEurl (std::shared_ptr<const sh::filesystem::Eurl> eurl,  
                      sh::filesystem::Operation *op, QVariant value)
```

Applies the detail value to a given eurl (without using any caches).

Used for transferring some details when a file transfer occurs.

Optionally implement the one of the applyValueBy... methods and register the instance with *sh::filesystem::FilesystemOperation::addTransferrableDetailColumn*. For performance reasons, you should decide to implement applyValueByEurl.

```
void applyValueByNode (std::shared_ptr<FilesystemNode> node, sh::filesystem::Operation  
                      *op, QVariant value)
```

Applies the detail value to a given node (without using any caches).

Used for transferring some details when a file transfer occurs.

Optionally implement the one of the applyValueBy... methods and register the instance with *sh::filesystem::FilesystemOperation::addTransferrableDetailColumn*. For performance reasons, you should decide to implement applyValueByEurl.

## Public Members

```
std::function<QString (std::shared_ptr<sh::filesystem::FilesystemNode>,  
                      sh::filesystem::Operation*)> _determineValue
```

```
std::function<void (std::shared_ptr<sh::filesystem::Eurl>, sh::filesystem::Operation*, QString)>  
_applyValue
```

## Public Static Functions

```
QVariant VALUE_UNAVAILABLE ()
```

A special value expressing that the value is not yet available.

```
void registerKnownDetailColumn (QString name, std::shared_ptr<DetailColumn> col-  
                               umn)
```

```
std::shared_ptr<DetailColumn> findKnownDetailColumn (QString name)
```

## Public Static Attributes

```
const uint DISPLAYINDEX_CORE = 10000
```

```
const uint DISPLAYINDEX_INTERESTING = 20000
```

```
const uint DISPLAYINDEX_EXOTIC = 30000
```

```
class ApiFilePropertyDialogTab: public sh::ui::FilePropertyDialogTab  
    #include <apifilepropertydialogtab.h>
```

## Public Functions

**ApiFilePropertyDialogTab** (QString *title*, QList<PropertyConfig> *properties*)

QString **title** () **override**

The tab title. .

QList<QString> **properties** () **override**

Returns the list of property labels (one entry for each widget).

Each property is displayed with a header and a widget (created in createWidget) with some content (populated in updateWidget).

void **populateWidget** (int *i*, sh::ui::FilePropertyDialogTabActionsView \**widget*) **override**

Populates the tab widget for the i-th property.

Typically uses the FilePropertyDialog::create\* methods to create sub-widgets.

See also *dialog*()).

void **updateWidget** (int *i*, sh::ui::FilePropertyDialogTabActionsView \**widget*, sh::filesystem::Operation \**op*) **override**

Populates the widget for the i-th property with actual content. .

QString **titleWithoutMnemonic** ()

The tab title without mnemonic.

void **refresh** ()

Refreshes the complete content.

Typically called after some user actions in a property widget require that all the other content must be refreshed.

QList<std::shared\_ptr<sh::filesystem::FilesystemNode>> **nodes** ()

The nodes to show.

FilePropertyDialogTabActionsView \***widgetAt** (int *i*)

Returns the widget for the i-th property.

int **widgetCount** ()

Returns the number of widgets.

std::shared\_ptr<FilePropertyDialog> **dialog** ()

Returns the associated dialog.

## Public Members

std::function< QList< QString >int, sh::filesystem::Operation \*, qintptr>> \_update

std::function<void (QString)> \_buttonTriggered

### Public Static Attributes

```
const int PROPERTY_TYPE_STRING = 1
const int PROPERTY_TYPE_STRINGMAP = 2
const int PROPERTY_TYPE_ICONTEXTBANNER = 3
```

### Private Types

```
typedef QPair<QString, QString> StringPair
```

### Private Members

```
QString _title
QList<PropertyConfig> _properties
class PropertyConfig
    #include <apifilepropertydialogtab.h>
```

### Public Types

```
typedef QPair<QString, QString> ButtonConfig
```

### Public Functions

```
PropertyConfig (QString title, int propertytype, QList<ButtonConfig> buttons)
```

### Public Members

```
QString title
int propertytype
QList<ButtonConfig> buttons
```

```
class ApiFilePropertyDialogTabFactory : public sh::ui::FilePropertyDialogTabFactory
    #include <apifilepropertydialogtab.h>
```

### Public Functions

```
ApiFilePropertyDialogTabFactory ()
std::shared_ptr<sh::ui::FilePropertyDialogTab> construct (std::shared_ptr<sh::ui::FilePropertyDialog>
                                                         dialog)
std::shared_ptr<sh::ui::FilePropertyDialogTab> construct (std::shared_ptr<sh::ui::FilePropertyDialog>
                                                         dialog) = 0
    Constructs a fresh instance of a FilePropertyDialogTab implementation. .
```

## Public Members

```
std::function< std::shared_ptr< sh::scripting::api::ApiFilePropertyDialogTab >>
```

## Public Static Attributes

```
const int REGISTER_FILEPROPERTYDIALOGTAB_INDEX_CORE = 10000
```

Base value for display indexes of core (i.e. maximum important) tabs.

Used in *FilePropertyDialog::addTabFactory*.

```
const int REGISTER_FILEPROPERTYDIALOGTAB_INDEX_VERYIMPORTANT = 20000
```

Base value for display indexes of very important tabs.

Used in *FilePropertyDialog::addTabFactory*.

```
const int REGISTER_FILEPROPERTYDIALOGTAB_INDEX_IMPORTANT = 30000
```

Base value for display indexes of important tabs.

Used in *FilePropertyDialog::addTabFactory*.

```
const int REGISTER_FILEPROPERTYDIALOGTAB_INDEX_NORMAL = 40000
```

Base value for display indexes of tabs with medium importance.

Used in *FilePropertyDialog::addTabFactory*.

```
const int REGISTER_FILEPROPERTYDIALOGTAB_INDEX_EXOTIC = 50000
```

Base value for display indexes of exotic (i.e. less important) tabs.

Used in *FilePropertyDialog::addTabFactory*.

```
class ApiFilesystemHandler : public sh::filesystem::FilesystemHandler
#include <apifilesystemhandler.h>
```

## Public Functions

```
ApiFilesystemHandler (sh::filesystem::FilesystemModel *model)
```

```
void itemList (sh::filesystem::Operation *op, std::shared_ptr<const
sh::filesystem::Eurl> eurl, sh::filesystem::FilesystemNodeType type,
sh::filesystem::FilesystemNodeListEditor *list)
```

```
void configureItems (sh::filesystem::Operation *op, QList<std::shared_ptr<sh::filesystem::FilesystemNode>>
nodes)
```

```
sh::filesystem::FilesystemNodeType getType (sh::filesystem::Operation *op,
std::shared_ptr<const sh::filesystem::Eurl>
eurl)
```

```
qint64 getSize (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
eurl)
```

```
QDateTime getMtime (sh::filesystem::Operation *op, std::shared_ptr<const
sh::filesystem::Eurl> eurl)
```

```
void setMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
eurl, QDateTime time)
```

```
QString getMimeType (sh::filesystem::Operation *op, std::shared_ptr<const
sh::filesystem::Eurl> eurl)
```

```
QString getLinkTarget (sh::filesystem::Operation *op, std::shared_ptr<const
sh::filesystem::Eurl> eurl)
```



```

QList<QString> listExtendedAttributes (sh::filesystem::Operation *op,
                                         std::shared_ptr<const sh::filesystem::Eurl>
                                         eurl)

quint64 getExtendedAttributeSize (sh::filesystem::Operation *op,
                                   std::shared_ptr<const sh::filesystem::Eurl>
                                   eurl, QString attribute)

QByteArray getExtendedAttribute (sh::filesystem::Operation *op,
                                  std::shared_ptr<const sh::filesystem::Eurl>
                                  eurl, QString attribute)

void setExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
                           sh::filesystem::Eurl> eurl, QString attribute, QByteArray
                           value)

void removeExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
                              sh::filesystem::Eurl> eurl, QString attribute)

QMap<QString, QString> getCustomAttributes (sh::filesystem::Operation
                                              *op, std::shared_ptr<const
                                              sh::filesystem::Eurl> eurl)

void setCustomAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
                          sh::filesystem::Eurl> eurl, QString attribute, QString value)

bool canGetFileContent (sh::filesystem::Operation *op, std::shared_ptr<const
                         sh::filesystem::Eurl> eurl)

std::shared_ptr<QIODevice> getFileContent (sh::filesystem::Operation
                                           *op, std::shared_ptr<const
                                           sh::filesystem::Eurl> eurl)

bool canCreateDirectory (sh::filesystem::Operation *op, std::shared_ptr<const
                          sh::filesystem::Eurl> eurl)

void createDirectory (sh::filesystem::Operation *op, std::shared_ptr<const
                       sh::filesystem::Eurl> eurl)

bool canCreateLink (sh::filesystem::Operation *op, std::shared_ptr<const
                     sh::filesystem::Eurl> eurl)

void createLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                  eurl, QString target)

bool canCreateFile (sh::filesystem::Operation *op, std::shared_ptr<const
                     sh::filesystem::Eurl> eurl)

void createFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                  eurl, QIODevice *content, HandlerTransfer *handlertransfer)

bool canDeleteItem (sh::filesystem::Operation *op, std::shared_ptr<const
                     sh::filesystem::Eurl> eurl)

void deleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                  eurl)

bool canRenameItem (sh::filesystem::Operation *op, std::shared_ptr<const
                     sh::filesystem::Eurl> src)

void renameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                  src, QString destpath)

QList<std::shared_ptr<sh::actions::AbstractActionItem>> getActions (const
                                                                    QList<std::shared_ptr<const
                                                                    sh::filesystem::Eurl>>
                                                                    eurls)

```

```
void itemList (sh::filesystem::Operation *op, std::shared_ptr<const
               sh::filesystem::Eurl> eurl, sh::filesystem::FilesystemNodeType type,
               sh::filesystem::FilesystemNodeListEditor *list) = 0
    Determine a list of subelements in a certain directory.

void configureItems (sh::filesystem::Operation *op, QList<std::shared_ptr<FilesystemNode>>
                    nodes)
    Configure newly created nodes (e.g. setting another icon or changing the display name).

sh::filesystem::FilesystemNodeType getType (sh::filesystem::Operation *op,
                                             std::shared_ptr<const sh::filesystem::Eurl>
                                             eurl) = 0
    Determine if a file is regular, or a dir, or a link, ...

qint64 getSize (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                eurl) = 0
    Determine the size of a file.

QDateTime getMtime (sh::filesystem::Operation *op, std::shared_ptr<const
                   sh::filesystem::Eurl> eurl) = 0
    Determine the mtime of a file.

void setMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
               eurl, QDateTime time) = 0
    Set the mtime of a file.

QString getMimeType (sh::filesystem::Operation *op, std::shared_ptr<const
                    sh::filesystem::Eurl> eurl) = 0
    Determine mime type of a file.

QString getLinkTarget (sh::filesystem::Operation *op, std::shared_ptr<const
                      sh::filesystem::Eurl> eurl) = 0
    Resolve a link.

QList<QString> listExtendedAttributes (sh::filesystem::Operation *op,
                                       std::shared_ptr<const sh::filesystem::Eurl>
                                       eurl)
    Fetches the list of available extended attributes for an entry.

quint64 getExtendedAttributeSize (sh::filesystem::Operation *op,
                                   std::shared_ptr<const sh::filesystem::Eurl>
                                   eurl, QString attribute)
    Returns the size of value for one extended attribute for an entry.

QByteArray getExtendedAttribute (sh::filesystem::Operation *op,
                                  std::shared_ptr<const sh::filesystem::Eurl>
                                  eurl, QString attribute)
    Returns the value for one extended attribute for an entry.

void setExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
                           sh::filesystem::Eurl> eurl, QString attribute, QByteArray
                           value)
    Sets the value for one extended attribute for an entry.

void removeExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
                              sh::filesystem::Eurl> eurl, QString attribute)
    Removes one extended attributes for an entry.

QMap<QString, QString> getCustomAttributes (sh::filesystem::Operation
                                             *op, std::shared_ptr<const
                                             sh::filesystem::Eurl> eurl)
    Returns the custom attributes for an entry.
```

In contrast to extended attributes, the custom attributes are not directly stored in the filesystem this way, but handle implementation specific aspects (like permissions).

void **setCustomAttribute** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QString attribute, QString value)  
Sets the value for one custom attribute for an entry.

See also getCustomAttributes.

bool **canGetFileContent** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0  
Returns if it is allowed to get the content of a certain file.

std::shared\_ptr<QIODevice> **getFileContent** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0  
Get the content of a file.

bool **canCreateDirectory** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0  
Returns if it is allowed to create subdirectories in a certain directory.

void **createDirectory** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0  
Create a directory.

bool **canCreateLink** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0  
Returns if it is allowed to create a link in a certain directory.

void **createLink** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QString target) = 0  
Create a link.

bool **canCreateFile** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0  
Returns if it is allowed to create files in a certain directory.

void **createFile** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QIODevice \*content, HandlerTransfer \*handlertransfer) = 0  
Creates a file with some content.

It may use handlertransfer for some better integration, like cancel support and progress monitoring.

bool **canDeleteItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0  
Returns if it is allowed to delete a certain file/directory/link/...

void **deleteItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0  
Delete a file/directory/link/...

void **deleteDirectoryIfEmpty** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)  
Deletes a directory only if it is empty.

bool **canRenameItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> src) = 0  
Returns if it is allowed to rename a certain file, directory, link, ... (see renameItem).

void **renameItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> src, QString destpath) = 0  
Renames an item to another path.

It can be a simple filename change or also changes in the deeper path segments.

```
QList<std::shared_ptr<sh::actions::AbstractActionItem>> getActions (const
                                                                    QList<std::shared_ptr<const
                                                                    sh::filesystem::Eurl>>
                                                                    eurls) = 0
```

Returns a list of actions (see former example) for showing for certain files.

```
bool requiresResolvedLinks ()
```

Returns if this handler requires to be used by the engine with resolved links.

```
void viewEnteredDirectory (std::shared_ptr<const sh::filesystem::Eurl>)
```

Callback for preparing stuff when the view entered some directory (e.g. for enabling watchers).  
See also viewLeftDirectory.

```
void viewLeftDirectory (std::shared_ptr<const sh::filesystem::Eurl>)
```

Callback for preparing stuff when the view left some directory. See also viewEnteredDirectory.

```
bool isWellKnownScheme ()
```

Returns if the scheme for this handler is a well known one (which external programs typically would understand).

```
void search (sh::filesystem::Operation *op, std::shared_ptr<const
sh::filesystem::Eurl> eurl, std::function<void> std::shared_ptr<const
sh::filesystem::Eurl>, QList<std::shared_ptr<sh::search::SearchCriterion>>,
sh::filesystem::FilesystemNodeType ftype
> addfile, std::function<void>std::shared_ptr<const sh::filesystem::Eurl>> visitdir,
QList<std::shared_ptr<sh::search::SearchCriterion>> criteriaHelps searching for files.
```

Override this method with a behavior identical to the default, if a specialized implementation can be much faster than the default one.

```
void customizeUi (std::shared_ptr<sh::filesystem::FilesystemNode> node, bool *showSearch-
Panel)
```

Creates a custom gui, which becomes part of the fileview.

```
sh::filesystem::FilesystemModel *model ()
```

A convenience shortcut to the *sh::filesystem::FilesystemModel* (a singleton).

## Public Members

```
std::function<int (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl)>
_getType
```

```
std::function<qint64 (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl)>
_getSize
```

```
std::function<double (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl>
eurl)> _getMtime
```

```
std::function<void (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl,
double)> _setMtime
```

```
std::function<QString (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl>
eurl)> _getLinkTarget
```

```
std::function< QList< QString >sh::filesystem::Operation *op, std::shared_ptr< s
```

```
std::function<quint64 (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl,
QString attribute)> _getExtendedAttributeSize
```

```
std::function<QByteArray (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl>
eurl, QString attribute)> _getExtendedAttribute
```

```

std::function<void (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl,
    QString attribute, QByteArray value) > _setExtendedAttribute
std::function<void (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl,
    QString attribute) > _removeExtendedAttribute
std::function< QList< QString > sh::filesystem::Operation *op, std::shared_ptr< sh::filesystem::Eurl> eurl >
    _getCustomAttributes
std::function<void (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl,
    QString attribute, QString value) > _setCustomAttribute
std::function<QString (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl) > _getMimeType
std::function<bool (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl) > _canCreateDirectory
std::function<void (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl) > _createDirectory
std::function<bool (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl) > _canCreateLink
std::function<void (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl,
    QString target) > _createLink
std::function<bool (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl) > _canCreateFile
std::function<bool (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl) > _canDeleteItem
std::function<void (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl) > _deleteItem
std::function<bool (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> src) > _canRenameItem
std::function<void (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> src,
    QString destpath) > _renameItem
std::function<void (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl, int
    type, sh::filesystem::FilesystemNodeListEditor *list) > _itemlist
std::function<void (sh::filesystem::Operation *op, QList<std::shared_ptr<sh::filesystem::FilesystemNode>>) >
    _configureItems
std::function<bool (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl) > _canGetFileContent
std::function< std::shared_ptr< QIODevice > sh::filesystem::Operation *op, std::shared_ptr< sh::filesystem::Eurl> eurl >
    _getFileContent
std::function<void (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl,
    QIODevice *content, HandlerTransfer *handlertransfer) > _createFile
std::function< QList< std::shared_ptr< sh::actions::AbstractActionItem > > > _getActions
class ApiFilesystemOperationProgressMonitor : public sh::filesystem::FilesystemOperationProgressMonitor
#include <apifilesystemoperationprogressmonitor.h>

```

## Public Functions

**ApiFilesystemOperationProgressMonitor** (*sh::actions::ActionExecutionInfo* \*actionExecution)

bool **hasItemInfo** ()

Checks if this progress monitor provides information about how many items of a certain total number are transferred so far.

quint64 **doneItems** ()

Checks how many items are transferred so far.

quint64 **allItems** ()

Checks how many items are to be transferred in total (as predicted in the current moment).

bool **hasBytesInfo** ()

Checks if this progress monitor provides information about how many byte of a certain total number are transferred so far.

quint64 **doneBytes** ()

Checks how many bytes are transferred so far.

quint64 **allBytes** ()

Checks how many bytes are to be transferred in total (as predicted in the current moment).

QString **getItemInfoFrom** ()

Returns the current source of transfer (as textual information).

QString **getItemInfoTo** ()

Returns the current destination of transfer (as textual information).

QString **estimation** ()

Returns the current performance and time estimation (as textual information).

## Public Members

std::function<void ()> **\_changed**

std::function<bool (QList<*sh::filesystem::FilesystemOperationTransfers::OperationStep\**>)>  
**\_resolveConflicts**

**class ApiGlobalObject : public *sh::base::Singleton***  
*#include <apiglobalobject.h>*

## Public Functions

std::shared\_ptr<const *sh::filesystem::Eurl*> **getEurlFromString** (QString *eurl*)

void **logDebug** (QString *msg*)

void **logInfo** (QString *msg*)

void **logWarning** (QString *msg*)

void **logError** (QString *msg*)

int **filesystemnodetype\_file** ()

int **filesystemnodetype\_directory** ()

int **filesystemnodetype\_link** ()

```

int filesystemnodetype_unknown()
int filesystemnodetype_firsttype()
int filesystemnodetype_lastphysicaltype()
int filesystemnodetype_none()
int filesystemnodetype_specialtreeonlydirectory()
int filesystemnodetype_lasttype()
int actiondefaultprecedencevalues_openfile()
int actiondefaultprecedencevalues_builtinspecialopen()
int messageboxbutton_ok()
int messageboxbutton_continue()
int messageboxbutton_cancel()
int messageboxbutton_retry()
int messageboxbutton_yes()
int messageboxbutton_no()
int displayindex_core()
int displayindex_interesting()
int displayindex_exotic()
int configurationcategory_gui()
int configurationcategory_behavior()
int configurationcategory_externaltools()
int settinggroup_global()
int settinggroup_gui()
int settinggroup_filehandling()
int settinggroup_dataview()
int settinggroup_behavior()
int settinggroup_special()
int paneldetail_positionindex_veryinteresting()
int paneldetail_positionindex_interesting()
int paneldetail_positionindex_exotic()
int operationstep_conflictresolution_mergeddirectories()
int operationstep_conflictresolution_overwritedestination()
int operationstep_conflictresolution_renamedestinationbefore()
int operationstep_conflictresolution_skip()
int operationstep_conflictresolution_unresolved()
int operationstep_conflictresolution_indirect()
int operationstep_conflictresolution_usedifferentdestinationname()

```

```
int filepropertydialogtab_index_core ()
int filepropertydialogtab_index_veryimportant ()
int filepropertydialogtab_index_important ()
int filepropertydialogtab_index_normal ()
int filepropertydialogtab_index_exotic ()
int filepropertydialogtab_propertytype_string ()
int filepropertydialogtab_propertytype_stringmap ()
int filepropertydialogtab_propertytype_icontextbanner ()
int searchcriterionfactory_index_core ()
int searchcriterionfactory_index_normal ()
int searchcriterionfactory_index_exotic ()
int thumbnailprovider_index_core ()
int thumbnailprovider_index_normal ()
int thumbnailprovider_index_exotic ()
int thumbnailprovider_index_fallback ()
void register_predicatedactionfactory (std::shared_ptr<sh::scripting::api::ApiActionFactory>
                                     f)

bool isDebugBuild ()

std::shared_ptr<sh::scripting::api::ApiFilesystemHandler> create_FilesystemHandler ()
std::shared_ptr<sh::scripting::api::ApiActionActionItem> create_ActionActionItem (QString
                                     text,
                                     bool
                                     en-
                                     abled,
                                     QString
                                     icon,
                                     int
                                     de-
                                     fault-
                                     Ac-
                                     tion-
                                     Prece-
                                     dence)
```



```

std::shared_ptr<sh::scripting::api::ApiSubmenuItem> create_SubmenuItem (QString
                                                                    text,
                                                                    bool
                                                                    en-
                                                                    abled,
                                                                    QString
                                                                    icon,
                                                                    int
                                                                    de-
                                                                    fault-
                                                                    Ac-
                                                                    tion-
                                                                    Prece-
                                                                    dence)

std::shared_ptr<sh::scripting::api::ApiDetailColumn> create_DetailColumn (QString
                                                                    name,
                                                                    QString
                                                                    display-
                                                                    Name,  int
                                                                    displayIn-
                                                                    dex,  bool
                                                                    sort_doTypediff,
                                                                    int default-
                                                                    Width, bool
                                                                    isRightAl-
                                                                    igned)

std::shared_ptr<sh::scripting::api::ApiFilePropertyDialogTab> create_FilePropertyDialogTab (QString
                                                                    ti-
                                                                    tle,
                                                                    QList<QStri
                                                                    prop-
                                                                    er-
                                                                    ties)

std::shared_ptr<sh::scripting::api::ApiFilePropertyDialogTabFactory> create_FilePropertyDialogTabFacto
std::shared_ptr<sh::scripting::api::ApiActionFactory> create_ActionFactory (QString
                                                                    category,
                                                                    QList<std::shared_ptr<actions::P
                                                                    predi-
                                                                    cates)

std::shared_ptr<sh::scripting::api::ApiPanelDetailFactorySingle> create_DetailFactorySingle (int
                                                                    po-
                                                                    si-
                                                                    tion,
                                                                    int
                                                                    val-
                                                                    ueWidth-
                                                                    Hint)

```

```
std::shared_ptr<sh::scripting::api::ApiPanelDetailFactoryMulti> create_DetailFactoryMulti (int
                                                                    po-
                                                                    si-
                                                                    tion,
                                                                    int
                                                                    val-
                                                                    ueWidth-
                                                                    Hint)

std::shared_ptr<sh::scripting::api::ApiFilesystemOperationProgressMonitor> create_FilesystemOperationP

std::shared_ptr<sh::filesystem::FilesystemNode> _createFilesystemNode (std::shared_ptr<sh::filesystem::Eurl>
                                                                    eurl,
                                                                    sh::scripting::api::ApiFilesystemHan
                                                                    *handler,
                                                                    int nodeType,
                                                                    std::shared_ptr<sh::filesystem::Filesy
                                                                    parent, bool
                                                                    doinsert, bool
                                                                    showInitial-
                                                                    LoadingLabel,
                                                                    bool isHid-
                                                                    den)

std::shared_ptr<sh::scripting::api::ApiThumbnailProvider> create_ThumbnailProvider ( )

std::shared_ptr<sh::filesystem::FilesystemNode> _getOrCreateFilesystemNode (std::shared_ptr<sh::filesystem
                                                                    eurl,
                                                                    sh::scripting::api::ApiFilesystem
                                                                    *han-
                                                                    dler,
                                                                    int
                                                                    node-
                                                                    type,
                                                                    std::shared_ptr<sh::filesystem
                                                                    parent,
                                                                    bool
                                                                    doin-
                                                                    sert,
                                                                    bool
                                                                    showIni-
                                                                    tial-
                                                                    Load-
                                                                    ingLabel,
                                                                    bool
                                                                    isHid-
                                                                    den)

void _register_FilesystemHandler (std::shared_ptr<sh::scripting::api::ApiFilesystemHandler>
                                                                    handler, QString scheme)

std::shared_ptr<sh::filesystem::FilesystemNode> modelRootNode ( )
```

```

QList<std::shared_ptr<sh::filesystem::FileSystemNode>> _findFileSystemNodesForEurl (std::shared_ptr<const sh::filesystem::Eurl> eurl)

QList<std::shared_ptr<sh::filesystem::FileSystemNode>> _tryGetFileSystemNodesForEurl (std::shared_ptr<const sh::filesystem::Eurl> eurl)

void refreshData (std::shared_ptr<const sh::filesystem::Eurl> eurl, bool forceFindParent, bool withDetails)

std::shared_ptr<filesystem::Eurl> createEurl (QString scheme, QString hostname, QString path)

std::shared_ptr<sh::filesystem::Operation> createOperation ()

sh::scripting::StringMap<QString> createArgumentException ()

sh::scripting::StringMap<QString> createIOException ()

sh::scripting::StringMap<QString> createRuntimeException ()

sh::scripting::StringMap<QString> createProgramException ()

sh::scripting::StringMap<QString> createException ()

std::shared_ptr<sh::scripting::api::ApiReadDataDevice> _createReadDataDevice ()

std::shared_ptr<ApiSetting> createSetting (QString name, QString description, int group, bool isAdvancedSetting, bool isGlobal, bool isPerFileview)

std::shared_ptr<sh::scripting::api::ApiThread> create_Thread ()

std::shared_ptr<sh::scripting::api::ApiTimer> create_Timer ()

std::shared_ptr<sh::scripting::api::ApiSearchCriterion> create_SearchCriterion (std::shared_ptr<sh::scripting::factory> factory)

std::shared_ptr<sh::scripting::api::ApiSearchCriterionFactory> create_SearchCriterionFactory (QString key, QString description)

sh::ui::MainWindow *mainwindow ()

void registerPanelDetailFactorySingle (std::shared_ptr<sh::scripting::api::ApiPanelDetailFactorySingle>)

void registerPanelDetailFactoryMulti (std::shared_ptr<sh::scripting::api::ApiPanelDetailFactoryMulti>)

void registerSetting (std::shared_ptr<sh::settings::Setting> setting)

void openItems (QList<std::shared_ptr<const sh::filesystem::Eurl>> eurls)

void registerFilePropertyDialogTabFactory (int index, std::shared_ptr<sh::scripting::api::ApiFilePropertyDialogTabFactory>)

void registerThumbnailProvider (int idx, std::shared_ptr<sh::scripting::api::ApiThumbnailProvider> thumbnailprovider)

void registerSearchCriterionFactory (int idx, std::shared_ptr<ApiSearchCriterionFactory> apicrit)

```

```
std::shared_ptr<sh::configuration::ConfigurationValue> register_ConfigurationValueInt (QString
                                                                    name,
                                                                    int
                                                                    de-
                                                                    ft,
                                                                    QString
                                                                    desc,
                                                                    int
                                                                    cat-
                                                                    e-
                                                                    gory,
                                                                    QString
                                                                    longdesc,
                                                                    QString
                                                                    change-
                                                                    hint)
```

```
std::shared_ptr<sh::configuration::ConfigurationValue> register_ConfigurationValueFloat (QString
                                                                    name,
                                                                    int
                                                                    de-
                                                                    ft,
                                                                    QString
                                                                    desc,
                                                                    int
                                                                    cat-
                                                                    e-
                                                                    gory,
                                                                    QString
                                                                    longdesc,
                                                                    QString
                                                                    change-
                                                                    hint)
```

```
std::shared_ptr<sh::configuration::ConfigurationValue> register_ConfigurationValueBool (QString
                                                                    name,
                                                                    bool
                                                                    de-
                                                                    ft,
                                                                    QString
                                                                    desc,
                                                                    int
                                                                    cat-
                                                                    e-
                                                                    gory,
                                                                    QString
                                                                    longdesc,
                                                                    QString
                                                                    change-
                                                                    hint)
```

```

std::shared_ptr<sh::configuration::ConfigurationValue> register_ConfigurationValueString (QString
                                                                    name,
                                                                    QString
                                                                    de-
                                                                    flt,
                                                                    QString
                                                                    desc,
                                                                    int
                                                                    cat-
                                                                    e-
                                                                    gory,
                                                                    QString
                                                                    longdesc,
                                                                    QString
                                                                    change-
                                                                    hint)

void registerTransferrableDetailColumn (int i, std::shared_ptr<sh::filesystem::DetailColumn>
                                                                    detailColumn)

std::shared_ptr<sh::filesystem::DetailColumn> findDetailColumnByName (QString
                                                                    name)

QString shallotDataDir ()
QString shallotBuiltinPluginDir ()
QString shallotSystemPluginDir ()
QString shallotUserPluginDir ()
QString shallotLanguage ()

sh::tools::BookmarkManager *bookmarkManager ()

std::shared_ptr<sh::actions::Predicate> create_OnDirectoriesPredicate ()
std::shared_ptr<sh::actions::Predicate> create_OnFilesPredicate ()
std::shared_ptr<sh::actions::Predicate> create_OnLinksPredicate ()
std::shared_ptr<sh::actions::Predicate> create_OnSingleEntrySelectionPredicate ()
std::shared_ptr<sh::actions::Predicate> create_DontResolveLinksPredicate ()
std::shared_ptr<sh::actions::Predicate> create_HideOnCurrentDirectoryLevelPredicate ()
std::shared_ptr<sh::actions::Predicate> create_HideOnSelectionLevelPredicate ()
std::shared_ptr<sh::actions::Predicate> create_ByRegExpPredicate (QString pattern)
std::shared_ptr<sh::actions::Predicate> create_KeyShortcutPredicate (QString short-
                                                                    cut,      bool
                                                                    triggersOn-
                                                                    CurrentDirec-
                                                                    toryLevel)

std::shared_ptr<sh::actions::Predicate> create_PositionIndexPredicate (int index)

void doInitialize ()
    Executes singleton initialization.

void doShutdown ()
    Executes singleton shutdown.

```

```
void shutdown ()  
    Shutdown down this singleton.  
  
bool isAlive ()  
    Returns if this singleton is alive (true until its shutdown begins).
```

### Private Functions

```
ApiGlobalObject ()  
  
class ApiHelperMethods  
    #include <apihelpermethods.h>
```

### Public Static Functions

```
QByteArray QIODevice_read (QIODevice *self)  
  
QByteArray QIODevice_readall (QIODevice *self)  
  
QByteArray ApiReadDataDevice_read (sh::scripting::api::ApiReadDataDevice *self)  
  
QByteArray ApiReadDataDevice_readall (sh::scripting::api::ApiReadDataDevice  
                                        *self)  
  
void FilesystemNodeList_resetItems (sh::filesystem::FilesystemNodeList  
                                     *self, int type,  
                                     QList<std::shared_ptr<sh::filesystem::FilesystemNode>>  
                                     list)  
  
void FilesystemNode_addDetail (filesystem::FilesystemNode *self,  
                               std::shared_ptr<sh::filesystem::DetailColumn> col)  
  
void FilesystemNode_setIcon (sh::filesystem::FilesystemNode *self, QString icon)  
  
void FilesystemNode_setDisplayName (filesystem::FilesystemNode *self, QString displayname)  
  
bool FilesystemNode_isHidden (filesystem::FilesystemNode *self)  
  
void FilesystemNode_setHidden (filesystem::FilesystemNode *self, bool v)  
  
void ConfigurationValue_setConfigValueInt (sh::configuration::ConfigurationValue  
                                             *self, int v)  
  
void ConfigurationValue_setConfigValueFloat (sh::configuration::ConfigurationValue  
                                              *self, double v)  
  
void ConfigurationValue_setConfigValueBool (sh::configuration::ConfigurationValue  
                                              *self, bool v)  
  
void ConfigurationValue_setConfigValueString (sh::configuration::ConfigurationValue  
                                              *self, QString v)  
  
QList<std::shared_ptr<sh::filesystem::Eurl>> FilesystemOperation_itemlist (sh::filesystem::FilesystemOperation  
                                                                              *self,  
                                                                              std::shared_ptr<const  
                                                                              sh::filesystem::Eurl>  
                                                                              eurl)
```

```

QList<std::shared_ptr<sh::filesystem::Eurl>> FilesystemOperation_itemlistByType (sh::filesystem::FilesystemOperation *self,
std::shared_ptr<const sh::filesystem::Eurl> eurl,
int ntype)

int FilesystemOperation_getType (sh::filesystem::FilesystemOperation *self,
std::shared_ptr<const sh::filesystem::Eurl> eurl)

void FilesystemOperation_createFile (sh::filesystem::FilesystemOperation *self,
std::shared_ptr<const sh::filesystem::Eurl> eurl,
sh::scripting::api::ApiReadDataDevice *content, std::shared_ptr<filesystem::FilesystemOperationProgressMonitor> progressmon)

QList<std::shared_ptr<sh::filesystem::FilesystemNode>> FilesystemOperation_resolveNodeLink (sh::filesystem::FilesystemNode *self,
std::shared_ptr<const sh::filesystem::FilesystemNode> node)

QList<std::shared_ptr<sh::filesystem::FilesystemNode>> FilesystemOperation_resolveNodeLink_NonRecursive (sh::filesystem::FilesystemNode *self,
std::shared_ptr<const sh::filesystem::FilesystemNode> node)

std::shared_ptr<sh::filesystem::Eurl> FilesystemOperation_resolveEurlLink (sh::filesystem::FilesystemOperation *self,
std::shared_ptr<const sh::filesystem::Eurl> eurl)

std::shared_ptr<sh::filesystem::Eurl> FilesystemOperation_resolveEurlLink_NonRecursive (sh::filesystem::FilesystemOperation *self,
std::shared_ptr<const sh::filesystem::Eurl> eurl)

void PanelDetail_setRow (sh::paneldetails::PanelDetail *self, QString key,
QList<QString> value)

void MainWindow_jumpToEurl (sh::ui::MainWindow *self, std::shared_ptr<const sh::filesystem::Eurl> eurl)

std::shared_ptr<sh::filesystem::FilesystemNode> MainWindow_getCurrentDirectoryNode (sh::ui::MainWindow *self)

void ApiActionActionItem_setEnabled (sh::scripting::api::ApiActionActionItem *self, bool enabled)

bool ApiActionActionItem_enabled (sh::scripting::api::ApiActionActionItem *self)

void ApiActionActionItem_setVisible (sh::scripting::api::ApiActionActionItem *self, bool visible)

bool ApiActionActionItem_visible (sh::scripting::api::ApiActionActionItem *self)

void ApiSubmenuItem_setSubitems (sh::scripting::api::ApiSubmenuItem *self, QList<std::shared_ptr<actions::AbstractActionItem>> subitems)

```

```
void ApiSubmenuItem_setEnabled (sh::scripting::api::ApiSubmenuItem  
                                *self, bool enabled)  
bool ApiSubmenuItem_enabled (sh::scripting::api::ApiSubmenuItem  
                              *self)  
void ApiSubmenuItem_setVisible (sh::scripting::api::ApiSubmenuItem  
                                *self, bool visible)  
bool ApiSubmenuItem_visible (sh::scripting::api::ApiSubmenuItem  
                              *self)  
int OperationStep_conflictResolution (filesystem::FilesystemOperationTransfers::OperationStep  
                                       *self)  
bool FilesystemOperationProgressMonitor_hasItemInfo (sh::scripting::api::ApiFilesystemOperationProgressMonitor  
                                                      *self)  
quint64 FilesystemOperationProgressMonitor_doneItems (sh::scripting::api::ApiFilesystemOperationProgressMonitor  
                                                       *self)  
quint64 FilesystemOperationProgressMonitor_allItems (sh::scripting::api::ApiFilesystemOperationProgressMonitor  
                                                      *self)  
bool FilesystemOperationProgressMonitor_hasBytesInfo (sh::scripting::api::ApiFilesystemOperationProgressMonitor  
                                                       *self)  
quint64 FilesystemOperationProgressMonitor_doneBytes (sh::scripting::api::ApiFilesystemOperationProgressMonitor  
                                                       *self)  
quint64 FilesystemOperationProgressMonitor_allBytes (sh::scripting::api::ApiFilesystemOperationProgressMonitor  
                                                      *self)  
QString FilesystemOperationProgressMonitor_getItemInfoFrom (sh::scripting::api::ApiFilesystemOperationProgressMonitor  
                                                             *self)  
QString FilesystemOperationProgressMonitor_getItemInfoTo (sh::scripting::api::ApiFilesystemOperationProgressMonitor  
                                                            *self)  
QString FilesystemOperationProgressMonitor_estimation (sh::scripting::api::ApiFilesystemOperationProgressMonitor  
                                                         *self)  
QList<std::shared_ptr<sh::filesystem::FilesystemNode>> ApiFilePropertyDialogTab_nodes (sh::scripting::api::ApiFilePropertyDialogTab  
                                                                                       *self)  
int ActionExecutionUserFeedback_messageBox (sh::actions::ActionExecutionUserFeedback  
                                             *self, QString m, QList<QString>  
                                             l, QString s, int i, int j)  
QString ActionExecutionUserFeedback_simpleInputDialog (sh::actions::ActionExecutionUserFeedback  
                                                         *self, QString text,  
                                                         QString default)  
int FilePropertyDialogTab_selectedIndexForProperty (ApiFilePropertyDialogTab  
                                                     *self, quintptr widget)  
void FilePropertyDialogTab_refresh (ApiFilePropertyDialogTab *self)  
class ApiPanelDetailFactoryMulti : public sh::paneldetails::PanelDetailFactoryForFunctionMulti  
    #include <apipaneldetailfactory.h>
```



## Public Functions

```
ApiPanelDetailFactoryMulti (int position, int valueWidthHint)

void setvalue (std::shared_ptr<sh::paneldetails::PanelDetail> detail,
               QList<std::shared_ptr<sh::filesystem::FilesystemNode>> nodes, filesystem::Operation *op)

void linktriggered (QList<std::shared_ptr<sh::filesystem::FilesystemNode>> nodes,
                    QString linktarget)

std::shared_ptr<PanelDetail> construct (QList<std::shared_ptr<sh::filesystem::FilesystemNode>>
                                       nodes, sh::filesystem::Operation *op)
```

## Public Members

```
std::function<void (sh::paneldetails::PanelDetail*, QList<std::shared_ptr<sh::filesystem::FilesystemNode>>,
                  sh::filesystem::Operation *op)> _setvalue

std::function<void (QList<std::shared_ptr<sh::filesystem::FilesystemNode>>,          QString) >
_linktriggered

class ApiPanelDetailFactorySingle : public sh::paneldetails::PanelDetailFactoryForFunctionSingle
#include <apipaneldetailfactory.h>
```

## Public Functions

```
ApiPanelDetailFactorySingle (int position, int valueWidthHint)

void setvalue (std::shared_ptr<sh::paneldetails::PanelDetail>,
               std::shared_ptr<sh::filesystem::FilesystemNode> node, filesystem::Operation *op)

void linktriggered (std::shared_ptr<sh::filesystem::FilesystemNode> node, QString linktarget)

std::shared_ptr<PanelDetail> construct (QList<std::shared_ptr<sh::filesystem::FilesystemNode>>
                                       nodes, sh::filesystem::Operation *op)
```

## Public Members

```
std::function<void (sh::paneldetails::PanelDetail*, std::shared_ptr<sh::filesystem::FilesystemNode>,
                  sh::filesystem::Operation *op)> _setvalue

std::function<void (std::shared_ptr<sh::filesystem::FilesystemNode>,          QString) >
_linktriggered

class ApiReadDataDevice : public sh::tools::ReadDataDevice
#include <apireaddatadevice.h>
```

## Public Functions

**ApiReadDataDevice** ()

QByteArray **getdata** ()

This method returns the next available chunk of data. It may return empty arrays whenever temporarily no data is available. Returning a null array (with `QByteArray::isNull() == true`) marks the end of the stream. .

void **\_ended** ()

## Public Members

std::function<QByteArray ()> **\_getdata**

## Private Members

bool **isended**

**class ApiSearchCriterion**: public *sh::search::criteria::AbstractActionDrivenSearchCriterion*  
#include <apisearchcriterion.h>

## Public Functions

**ApiSearchCriterion** (std::shared\_ptr<*sh::search::SearchCriterionFactory*> *factory*)

bool **match2** (*sh::filesystem::Operation* \**op*, std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*)

QList<QString> **getsearchspec** ()

bool **match** (*sh::filesystem::Operation* \**op*, std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*)

Checks if a given file (by *eurl*) matches this criterion. .

QString **valuedescription** ()

Returns a human readable text describing the current criterion shortly and precisely. .

QString **serialize** ()

Serializes this criterion to a string. See also *SearchCriterion::deserialize*.

std::shared\_ptr<*sh::search::SearchCriterionFactory*> **factory** ()

Returns the factory this criterion has created (provides some more metadata).

## Public Members

std::function<void (*sh::actions::ActionExecutionInfo*\*)> **\_configure**

std::function<bool (*sh::filesystem::Operation* \**op*, std::shared\_ptr<*sh::filesystem::Eurl*> *eurl*)>  
**\_match**

QStringList **searchspec**

## Public Static Functions

```
void _createEditor (std::shared_ptr<sh::search::SearchCriterion>                cfg,
                   sh::ui::SearchPanelConfiguration *panelcfg)
void _editorUpdateConfiguration (sh::ui::SearchPanelConfiguration *panelcfg,
                                std::shared_ptr<sh::search::SearchCriterion> c)

std::shared_ptr<sh::search::SearchCriterion> deserialize (QString s)
    Deserializes this criterion from a string. See also SearchCriterion::serialize.

class ApiSearchCriterionFactory : public sh::search::SearchCriterionFactory
    #include <apisearchcriterion.h>
```

## Public Functions

```
ApiSearchCriterionFactory (QString key, QString description)

QString key ()
    Returns the string key for the associated search criterion class. .

QString description ()
    Returns the description string for the associated search criterion class. .

void editor (std::shared_ptr<sh::search::SearchCriterion>                c,
            sh::ui::SearchPanelConfiguration *panelcfg)
void editorupdateconfig (sh::ui::SearchPanelConfiguration *panelcfg,
                       std::shared_ptr<sh::search::SearchCriterion> c)

std::shared_ptr<sh::search::SearchCriterion> construct ()
    Constructs an instance of the associated search criterion class. .

bool isVisibleFor (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::FilesystemNode>
                  node)
    Checks if the associated search criterion class should be offered to the user for a given node. .

void editor (std::shared_ptr<sh::search::SearchCriterion>                c,
            sh::ui::SearchPanelConfiguration *panelcfg) = 0
    Builds the search config ui in a search panel.

    Takes the data from c and populates panelcfg with it.

void editorupdateconfig (sh::ui::SearchPanelConfiguration *panelcfg,
                       std::shared_ptr<sh::search::SearchCriterion> c) = 0
    Updates own data with the content from the search config ui in a search panel.

    Takes the data from panelcfg and updates c with it.
```

## Public Members

```
std::function<QString (QList<QString>)> _getsearchspecdesc

std::function< std::shared_ptr< sh::search::SearchCriterion >> _construct

std::function<bool (sh::filesystem::Operation*, std::shared_ptr<sh::filesystem::FilesystemNode>)>
_isVisibleFor

class ApiSetting : public sh::settings::Setting
    #include <apisetting.h>
```

## Public Functions

**ApiSetting** (QString *name*, QString *description*, *sh::settings::SettingGroup* *group*, bool *isAdvancedSetting*, bool *isGlobal*, bool *isPerFileview*)

QString **name** ()  
Gets the internal name.

QString **description** ()  
Gets the description text.

*sh::settings::SettingGroup* **group** ()  
Gets the group;.

bool **isAdvancedSetting** ()  
Is this an advanced setting?

void **setValue** (QString *value*)  
Called from Shallot core when the value was set (for a not-per-fileview setting).

void **setValue** (*sh::ui::FileView* \**filelist*, QString *value*)  
Called from Shallot core when the value was set (for a per-fileview setting).

QString **getValue** (*sh::ui::FileView* \**filelist*)  
Get the currently set value.

bool **isGlobal** ()  
Does this setting apply globally or just for a certain subtree of nodes?

bool **isPerFileview** ()  
Does this setting apply for each fileview individually or for the complete main window?

QString **valueDescription** (QString *value*)  
Gets a human readable description text for a value.

## Public Members

std::function<void (QString)> **\_setValue1**

std::function<void (int, QString)> **\_setValue2**

std::function<QString (int)> **\_getValue**

std::function<QString (QString)> **\_valueDescription**

QString **\_name**

QString **\_description**

*sh::settings::SettingGroup* **\_group**

bool **\_isAdvancedSetting**

bool **\_isGlobal**

bool **\_isPerFileview**

## Public Static Functions

QString **getDescription** (int *g*)  
 Low-level function which gets the description text of a group.

```
class ApiSubmenuItem: public sh::actions::SubmenuItem
#include <apiaction.h>
```

## Public Functions

**ApiSubmenuItem** (QString *text*, bool *enabled*, QString *icon*, int *defaultActionPrecedence*)

void **initialize** () **override**  
 Initialize the action. This should make the time-consuming parts, e.g. for determining a label or enabled state.

const QList<std::shared\_ptr<sh::actions::AbstractActionItem>> **subitems** ()  
 Returns the list of subitems from this submenu.

void **setSubitems** (const QList<std::shared\_ptr<sh::actions::AbstractActionItem>> *subitems*)  
 Sets the subitems.

QString **text** ()  
 Returns the displayed text for this action.

QString **icon** ()  
 Returns the icon for this action.

bool **enabled** ()  
 Checks if this action is enabled.

bool **isChecked** ()  
 Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
 Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
 Returns the precedence for becoming the default action of a parent submenu.  
 This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
 Sets the displayed text.

void **setIcon** (QString *icon*)  
 Sets the icon.

void **setEnabled** (bool *enabled*)  
 Sets if the item is enabled.

void **setChecked** (bool *checked*)  
 Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
 Sets the visibility of this item.

`bool visible ()`  
Checks the visibility of this item (non-recursively).

`std::weak_ptr<AbstractActionItem> parentAction ()`  
Returns the parent action, if it is added to a container.

`void _setParentAction (std::shared_ptr<AbstractActionItem> parent)`  
Sets the parent action. .

`void initializeAsync (std::function<void>  
> oninitialized = 0)`Asynchronously initializes the action.

`void initializeSync ()`  
Synchronously initializes the action.

`bool isInitialized ()`  
Checks if this action is initialized.

`bool isInitializing ()`  
Checks if this action is initializing.

### Public Members

`std::function<void ()> _initialize`

### Signals

`void subitemsChanged ()`  
Emits when the list of subitems changed.

`void changed ()`  
Emits when some data changed.

**class ApiThread: public** `std::enable_shared_from_this<ApiThread>`  
*#include <apithread.h>*

### Public Functions

**ApiThread ()**  
**void start ()**

### Public Members

`std::function<void ()> _run`

**class ApiThumbnailProvider: public** *sh::tools::ThumbnailProvider*  
*#include <apithumbnailprovider.h>*

## Public Functions

**ApiThumbnailProvider** ()

void **getThumbnail** (*sh::filesystem::Operation* \*operation, std::shared\_ptr<*sh::filesystem::FileSystemNode*> node, QString contentType, int width, int height, QIcon \*icon)

## Public Members

std::function<QByteArray (*sh::filesystem::Operation*\*, std::shared\_ptr<*sh::filesystem::FileSystemNode*>, QString, int, int) > **\_getThumbnail**

**class ApiTimer**: public std::enable\_shared\_from\_this<*ApiTimer*>  
#include <apitimer.h>

## Public Functions

**ApiTimer** ()

void **start** (int interval)

void **stop** ()

## Public Members

std::function<void () > **\_run**

## Private Members

QMutex **\_mutex**

QTimer **\_timer**

bool **\_isexecuting** = false

QList<std::shared\_ptr<*ApiTimer*>> **\_runningTimers**

**namespace pythoninterop**

Interoperation layer between the Shallot core (C++) and Python plugins.

## Typedefs

**using StringMap** = QMap<QString, V>

## Functions

PyObject \***getPyNone** ()

bool **isPyNone** (PyObject \*o)

void \***getPointerFromShallotPyObject** (PyObject \*o)

std::shared\_ptr<void> **getSharedPointerFromShallotPyObject** (PyObject \*o)

PyObject \***getShallotPyObjectFromPointer** (void \*o, *sh::scripting::ApiClass* \*apiclass)

```
PyObject *getShallotPyObjectFromSharedPointer (std::shared_ptr<void> o,
                                              sh::scripting::ApiClass *apiclass)
sh::scripting::ApiClass *getApiClass (QString typeIdName)
PyObject *PyTupleGetItem (PyObject *o, int itm)
PyObject *PyListGetItem (PyObject *o, int itm)
int PyTupleCount (PyObject *o)
int PyListCount (PyObject *o)
bool PyIsTuple (PyObject *o)
bool PyIsList (PyObject *o)
bool PyIsUnicode (PyObject *o)
bool PyIsBytes (PyObject *o)
bool PyIsDict (PyObject *o)
bool PyIsBool (PyObject *o)
bool PyIsFloat (PyObject *o)
bool PyIsLong (PyObject *o)
void PyIncRef (PyObject *o)
void PyDecRef (PyObject *o)
void PyTupleSetItem (PyObject *o, int itm, PyObject *val)
PyObject *PyObjectCall (PyObject *fct, PyObject *args)
PyObject *PyTupleNew (int cnt)
PyObject *PyDictNew ()
PyObject *PyDictKeys (PyObject *o)
PyObject *PyDictGetItem (PyObject *dict, PyObject *key)
void PyDictSetItem (PyObject *dict, PyObject *key, PyObject *val)
bool PythonToC_ExecuteGuarded (std::function<void>
    > fct)
void invokePython (std::function<void>
    > fct)
template<typename R, typename T, typename ...ArgsF, typename ...ArgsT>
R CallNativeFunctionWithTuple (T *pObj, R (T::* f)) ArgsF...
    , std::tuple<ArgsT...> const &t
template<typename R, typename T, typename ...ArgsF, typename ...ArgsT>
R CallNativeHelperFunctionWithTuple (T *pObj, R (*f)) T*, ArgsF...
    , std::tuple<ArgsT...> const &t
template<int pos, class ...Args>
class _PyObjectFunctionToNativeFunction_SetTupleValue
    #include <pythonscriptinterpreter.h>
template<int pos>
class _PyObjectFunctionToNativeFunction_SetTupleValue<pos>
    #include <pythonscriptinterpreter.h>
```



## Public Static Functions

```
void setTupleValue (PyObject*)

template<int pos, class Arg, class ...Args>
class _PyObjectFunctionToNativeFunction_SetTupleValue<pos, Arg, Args...>
    #include <pythonscriptinterpreter.h>
```

## Public Static Functions

```
void setTupleValue (PyObject *tup, Arg arg, Args... args)

template<int pos, class ...Args>
class _PyObjectFunctionToNativeFunction_TupleValueSetter
    #include <pythonscriptinterpreter.h> This class is a workaround for gcc<4.9 bug 55914. It was not
    able to call ...::setTupleValue directly from within the invokePython lambda
```

## Public Functions

```
_PyObjectFunctionToNativeFunction_TupleValueSetter (Args... args)

void set (PyObject *tuple)
```

## Private Members

```
std::function<void (PyObject*)> _set
```

## Private Static Functions

```
void __set (Args... args, PyObject *tuple)

template<typename T, T>
struct CallNativeHelperMethodWithPyObjectArguments
    #include <pythonscriptinterpreter.h>

template<typename T, typename R, typename ... Args, R(*)(T *, Args...) MF> CallNativeHelperMethod
    #include <pythonscriptinterpreter.h>
```

## Public Static Functions

```
PyObject *call (PyObject *a, PyObject *b)

template<typename T, typename ... Args, void(*) (T *, Args...) MF> CallNativeHelperMethod
    #include <pythonscriptinterpreter.h>
```

### Public Static Functions

```
PyObject *call (PyObject *a, PyObject *b)

template<typename T, T>
struct CallNativeMethodWithPyObjectArguments
    #include <pythonscriptinterpreter.h>

template<typename T, typename R, typename ... Args, R(T::*)(Args...) const MF> *) (Args...
```

### Public Static Functions

```
PyObject *call (PyObject *a, PyObject *b)

template<typename T, typename R, typename ... Args, R(T::*)(Args...) MF> *) (Args...
```

### Public Static Functions

```
PyObject *call (PyObject *a, PyObject *b)

template<typename T, typename ... Args, void(T::*)(Args...) const MF> *) (Args...) c
    #include <pythonscriptinterpreter.h>
```

### Public Static Functions

```
PyObject *call (PyObject *a, PyObject *b)

template<typename T, typename ... Args, void(T::*)(Args...) MF> *) (Args...), MF >
    #include <pythonscriptinterpreter.h>
```

### Public Static Functions

```
PyObject *call (PyObject *a, PyObject *b)

class Converters
    #include <pythonscriptinterpreter.h>
```

### Public Static Functions

```
QString getStringFromPyObject (PyObject *o)
QVariant getVariantFromPyObject (PyObject *o)
int getIntFromPyObject (PyObject *o)
double getDoubleFromPyObject (PyObject *o)
qint64 getInt64FromPyObject (PyObject *o)
quint64 getUInt64FromPyObject (PyObject *o)
bool getBoolFromPyObject (PyObject *o)
QByteArray getByteArrayFromPyObject (PyObject *o)
```

```

void *getPointerFromPyObject (PyObject *o)
std::shared_ptr<void> getSharedPointerFromPyObject (PyObject *o)
PyObject *getPyObjectFromString (QString v)
PyObject *getPyObjectFromVariant (QVariant v)
PyObject *getPyObjectFromInt (int v)
PyObject *getPyObjectFromDouble (double v)
PyObject *getPyObjectFromInt64 (qint64 v)
PyObject *getPyObjectFromUInt64 (quint64 v)
PyObject *getPyObjectFromBool (bool v)
PyObject *getPyObjectFromByteArray (QByteArray v)

template<int mode, uint N>
struct FunctionBindTuple
    #include <pythonscriptinterpreter.h>

template<>
struct FunctionBindTuple<0, 0>
    #include <pythonscriptinterpreter.h>

```

### Public Static Functions

```

template<typename R, typename T, typename ...ArgsF, typename ...ArgsT, typename ...Args>
R applyTuple (T *pObj, R (T::* f)) ArgsF...
    , const std::tuple<ArgsT...>&, Args... args

template<uint N>
struct FunctionBindTuple<0, N>
    #include <pythonscriptinterpreter.h>

```

### Public Static Functions

```

template<typename R, typename T, typename ...ArgsF, typename ...ArgsT, typename ...Args>
R applyTuple (T *pObj, R (T::* f)) ArgsF...
    , const std::tuple<ArgsT...> &t, Args... args

template<>
struct FunctionBindTuple<1, 0>
    #include <pythonscriptinterpreter.h>

```

### Public Static Functions

```

template<typename R, typename T, typename ...ArgsF, typename ...ArgsT, typename ...Args>
R applyTuple (T *pObj, R (*f)) T*, ArgsF...
    , const std::tuple<ArgsT...>&, Args... args

template<uint N>
struct FunctionBindTuple<1, N>
    #include <pythonscriptinterpreter.h>

```

### Public Static Functions

```
template<typename R, typename T, typename ...ArgsF, typename ...ArgsT, typename ...Args>  
R applyTuple (T *pObj, R (*f)) T*, ArgsF...  
    , const std::tuple<ArgsT...> &t, Args... args
```

```
template<class V>  
class NativeToPyObject  
    #include <pythonscriptinterpreter.h> Helps to return a Python object from a native c++ value (with  
    reference ownership)
```

```
template<>  
class NativeToPyObject<bool>  
    #include <pythonscriptinterpreter.h>
```

### Public Static Functions

PyObject \*toPyObject (bool v)

```
template<>  
class NativeToPyObject<double>  
    #include <pythonscriptinterpreter.h>
```

### Public Static Functions

PyObject \*toPyObject (double v)

```
template<>  
class NativeToPyObject<int>  
    #include <pythonscriptinterpreter.h>
```

### Public Static Functions

PyObject \*toPyObject (int v)

```
template<>  
class NativeToPyObject<QByteArray>  
    #include <pythonscriptinterpreter.h>
```

### Public Static Functions

PyObject \*toPyObject (QByteArray v)

```
template<>  
class NativeToPyObject<qint64>  
    #include <pythonscriptinterpreter.h>
```

### Public Static Functions

```

PyObject *toPyObject (qint64 v)

template<typename V>
class NativeToPyObject<QList<V>>
    #include <pythonscriptinterpreter.h>

```

### Public Static Functions

```

PyObject *toPyObject (QList<V> v)

template<typename V>
class NativeToPyObject<QMap<QString, V>>
    #include <pythonscriptinterpreter.h>

```

### Public Static Functions

```

PyObject *toPyObject (QMap<QString, V> v)

template<>
class NativeToPyObject<QString>
    #include <pythonscriptinterpreter.h>

```

### Public Static Functions

```

PyObject *toPyObject (QString v)

template<>
class NativeToPyObject<qint64>
    #include <pythonscriptinterpreter.h>

```

### Public Static Functions

```

PyObject *toPyObject (qint64 v)

template<>
class NativeToPyObject<QVariant>
    #include <pythonscriptinterpreter.h>

```

### Public Static Functions

```

PyObject *toPyObject (QVariant v)

template<class V> shared_ptr< const V > >
    #include <pythonscriptinterpreter.h>

```

### Public Static Functions

```
PyObject *toPyObject (std::shared_ptr<const V> v)
template<class V> shared_ptr< V > >
#include <pythonscriptinterpreter.h>
```

### Public Static Functions

```
PyObject *toPyObject (std::shared_ptr<V> v)
template<class V>
class NativeToPyObject<V*>
#include <pythonscriptinterpreter.h>
```

### Public Static Functions

```
PyObject *toPyObject (V *v)
template<class R, class ...Args>
class PyObjectFunctionToNativeFunction
#include <pythonscriptinterpreter.h>
```

### Public Static Functions

```
std::function<R (Args...)> toNativeFunction
PyObject *self, PyObject *fct
template<class ...Args>
class PyObjectFunctionToNativeFunction<void, Args...>
#include <pythonscriptinterpreter.h>
```

### Public Static Functions

```
std::function<void (Args...)> toNativeFunction
PyObject *self, PyObject *fct
template<class V>
class PyObjectToNative
#include <pythonscriptinterpreter.h> Helps to return a c++ value from a Python object (borrows the
reference)
template<>
class PyObjectToNative<bool>
#include <pythonscriptinterpreter.h>
```

### Public Static Functions

```

bool toNative (PyObject*, PyObject *o)

template<>
class PyObjectToNative<double>
    #include <pythonscriptinterpreter.h>

```

### Public Static Functions

```

double toNative (PyObject*, PyObject *o)

template<>
class PyObjectToNative<int>
    #include <pythonscriptinterpreter.h>

```

### Public Static Functions

```

int toNative (PyObject*, PyObject *o)

template<>
class PyObjectToNative<QByteArray>
    #include <pythonscriptinterpreter.h>

```

### Public Static Functions

```

QByteArray toNative (PyObject*, PyObject *o)

template<>
class PyObjectToNative<qint64>
    #include <pythonscriptinterpreter.h>

```

### Public Static Functions

```

qint64 toNative (PyObject*, PyObject *o)

template<class V>
class PyObjectToNative<QList<V>>
    #include <pythonscriptinterpreter.h>

```

### Public Static Functions

```

QList<V> toNative (PyObject *a, PyObject *o)

template<>
class PyObjectToNative<QString>
    #include <pythonscriptinterpreter.h>

```

### Public Static Functions

```
QString toNative (PyObject*, PyObject *o)

template<>
class PyObjectToNative<quint64>
    #include <pythonscriptinterpreter.h>
```

### Public Static Functions

```
quint64 toNative (PyObject*, PyObject *o)

template<>
class PyObjectToNative<QVariant>
    #include <pythonscriptinterpreter.h>
```

### Public Static Functions

```
QVariant toNative (PyObject*, PyObject *o)

template<class V, class ... Args> function< V(Args...) > >
    #include <pythonscriptinterpreter.h>
```

### Public Static Functions

```
std::function<V (Args...) > toNative
    PyObject *self, PyObject *io

template<class V> shared_ptr< V > >
    #include <pythonscriptinterpreter.h>
```

### Public Static Functions

```
std::shared_ptr<V> toNative (PyObject*, PyObject *o)

template<class V>
class PyObjectToNative<StringMap<V>>
    #include <pythonscriptinterpreter.h>
```

### Public Static Functions

```
StringMap<V> toNative (PyObject *a, PyObject *o)

template<class V>
class PyObjectToNative<V*>
    #include <pythonscriptinterpreter.h>
```



### Public Static Functions

```

V *toNative (PyObject*, PyObject *o)

template<int pos, typename ...Args>
struct PyObjectTupleToNativeTuple
    #include <pythonscriptinterpreter.h>

template<int pos>
struct PyObjectTupleToNativeTuple<pos>
    #include <pythonscriptinterpreter.h>

```

### Public Static Functions

```

std::tuple toNativeTuple (PyObject*, PyObject*)

template<int pos, class T, class ...Args>
struct PyObjectTupleToNativeTuple<pos, T, Args...>
    #include <pythonscriptinterpreter.h>

```

### Public Static Functions

```

std::tuple<T, Args...> toNativeTuple (PyObject *a, PyObject *b)

template<typename D, typename T, T>
struct ScriptedMethodProxy
    #include <pythonscriptinterpreter.h>

template<typename T, typename R, typename ... Args, std::function< R(Args...)> T::*
    #include <pythonscriptinterpreter.h>

```

### Public Static Functions

```

void setImplementation (T *inst, std::function<R> Args...
    > vfcn

PyObject *pyObjectFunctionSetImplementation (PyObject *a, PyObject *b)

```

## 10.2.15 Namespace sh::scripting::api

**namespace** *sh::scripting::api*

Adapter functions and classes for interaction with scripting.

Some programming interfaces in the core are not directly usable for the scripting api. Scripting-friendly surrogates for them are here.

```

class ApiActionActionItem: public sh::actions::ActionActionItem
    #include <apiaction.h>

```

## Public Functions

**ApiActionActionItem** (QString *text*, bool *enabled*, QString *icon*, int *defaultActionPrecedence*)

void **\_execute** ()

void **\_initializeSync** ()

void **action** (*sh::actions::ActionExecutionInfo* \**info*) **override**

The action implementation, i.e. what the actions should actually do.

void **initialize** () **override**

Initialize the action. This should make the time-consuming parts, e.g. for determining a label or enabled state.

void **execute** ()

Executes this action.

void **execute** (*sh::actions::ActionExecutionInfo* \**info*)

Executes this action.

QKeySequence **shortcuthint** ()

Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()

Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)

Sets the keyboard shortcut for triggering this action.

QString **text** ()

Returns the displayed text for this action.

QString **icon** ()

Returns the icon for this action.

bool **enabled** ()

Checks if this action is enabled.

bool **isChecked** ()

Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()

Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**

Returns the precedence for becoming the default action of a parent submenu.

This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)

Sets the displayed text.

void **setIcon** (QString *icon*)

Sets the icon.

void **setEnabled** (bool *enabled*)

Sets if the item is enabled.

```

void setChecked (bool checked)
    Sets if the item is checked (has a cross in the ui).

void setVisible (bool visible)
    Sets the visibility of this item.

bool visible ()
    Checks the visibility of this item (non-recursively).

std::weak_ptr<AbstractActionItem> parentAction ()
    Returns the parent action, if it is added to a container.

void _setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.

```

## Public Members

```

std::function<void ()> _initialize

std::function<void (sh::actions::ActionExecutionInfo*)> _action

```

## Signals

```

void changed ()
    Emits when some data changed.

```

```

class ApiActionFactory : public sh::actions::AbstractActionFactory
    #include <apipredicatedactionfactory.h>

```

## Public Functions

```

ApiActionFactory (QString category, QList<std::shared_ptr<actions::Predicate>> predicates)

std::shared_ptr<sh::actions::AbstractActionItem> construct (QList<std::shared_ptr<sh::filesystem::FilesystemNode>>
    nodes)

std::shared_ptr<ActionInstantiation> actionAvailable (ActionInstantiation *instantiation)

```

## Public Members

```
std::function< std::shared_ptr< sh::actions::AbstractActionItem >QList< std::shared_ptr< sh::filesystem::DetailColumn >>>
class ApiDetailColumn: public sh::filesystem::DetailColumn
#include <apidetailcolumn.h>
```

## Public Functions

**ApiDetailColumn** (QString *name*, QString *displayName*, int *displayIndex*, bool *sort\_doTypediff*, int *defaultWidth*, bool *isRightAligned*)

QVariant **determineValue** (std::shared\_ptr<sh::filesystem::FilesystemNode> *node*, sh::filesystem::Operation \**op*)

void **applyValue** (std::shared\_ptr<const sh::filesystem::Eurl> *eurl*, sh::filesystem::Operation \**op*, QVariant *value*)

QString **name** ()  
The internal unique name.

QString **displayName** ()  
The display name.

bool **isValueAvailable** (std::shared\_ptr<FilesystemNode> *node*)  
Checks if a value for this detail is available for one given node.

QVariant **value** (std::shared\_ptr<FilesystemNode> *node*, bool *ignoreAged* = false)  
Returns the value for this detail for one given node.

### Parameters

- *ignoreAged*: If no value should be returned which is older than the last request (not useful for nearly all cases).

QString **displayValue** (std::shared\_ptr<FilesystemNode> *node*, const sh::filesystem::FilesystemModelFileviewProxy \**viewmodel*)  
Returns the stringified value for this details for one given node considering the configuration of a view.

bool **isVisible** ()  
If this detail shall be a visible column in the view.

QVariant **requestValue** (std::shared\_ptr<FilesystemNode> *node*, sh::filesystem::Operation \**op* = 0, bool *withNodeValues* = false, bool *withOperationsCache* = true)  
Requests to determine the value for this detail for one given node. Blocks until the value is available. Get it with the value method afterwards.

bool **sort\_doTypediff** ()  
If sorting should separate files and directories.

int **sort\_order** (std::shared\_ptr<FilesystemNode> *n1*, std::shared\_ptr<FilesystemNode> *n2*)  
The sorting order.

uint **displayIndex** ()  
Controls which place this column should get in the user interface.

QVariant **computeValue** (std::shared\_ptr<FilesystemNode> *node*, sh::filesystem::Operation \**op*)  
Returns a freshly determined value for this column and the given node. It neither asks nor populates any caches or storages.

int **defaultWidth** ()

bool **isRightAligned** ()

```
void applyValueByEurl (std::shared_ptr<const sh::filesystem::Eurl> eurl,
                      sh::filesystem::Operation *op, QVariant value)
```

Applies the detail value to a given eurl (without using any caches).

Used for transferring some details when a file transfer occurs.

Optionally implement the one of the `applyValueBy...` methods and register the instance with *sh::filesystem::FilesystemOperation::addTransferrableDetailColumn*. For performance reasons, you should decide to implement `applyValueByEurl`.

```
void applyValueByNode (std::shared_ptr<FilesystemNode> node, sh::filesystem::Operation *op,
                      QVariant value)
```

Applies the detail value to a given node (without using any caches).

Used for transferring some details when a file transfer occurs.

Optionally implement the one of the `applyValueBy...` methods and register the instance with *sh::filesystem::FilesystemOperation::addTransferrableDetailColumn*. For performance reasons, you should decide to implement `applyValueByEurl`.

## Public Members

```
std::function<QString (std::shared_ptr<sh::filesystem::FilesystemNode>,
                      sh::filesystem::Operation*)> _determineValue
```

```
std::function<void (std::shared_ptr<sh::filesystem::Eurl>, sh::filesystem::Operation*, QString)>
_applyValue
```

## Public Static Functions

```
QVariant VALUE_UNAVAILABLE ()
```

A special value expressing that the value is not yet available.

```
void registerKnownDetailColumn (QString name, std::shared_ptr<DetailColumn> column)
```

```
std::shared_ptr<DetailColumn> findKnownDetailColumn (QString name)
```

## Public Static Attributes

```
const uint DISPLAYINDEX_CORE = 10000
```

```
const uint DISPLAYINDEX_INTERESTING = 20000
```

```
const uint DISPLAYINDEX_EXOTIC = 30000
```

```
class ApiFilePropertyDialogTab: public sh::ui::FilePropertyDialogTab
    #include <apifilepropertydialogtab.h>
```

## Public Functions

**ApiFilePropertyDialogTab** (QString *title*, QList<PropertyConfig> *properties*)

QString **title** () **override**

The tab title. .

QList<QString> **properties** () **override**

Returns the list of property labels (one entry for each widget).

Each property is displayed with a header and a widget (created in `createWidget`) with some content (populated in `updateWidget`).

void **populateWidget** (int *i*, sh::ui::FilePropertyDialogTabActionsView \**widget*) **override**

Populates the tab widget for the *i*-th property.

Typically uses the `FilePropertyDialog::create*` methods to create sub-widgets.

See also `dialog()`.

void **updateWidget** (int *i*, sh::ui::FilePropertyDialogTabActionsView \**widget*,  
sh::filesystem::Operation \**op*) **override**

Populates the widget for the *i*-th property with actual content. .

QString **titleWithoutMnemonic** ()

The tab title without mnemonic.

void **refresh** ()

Refreshes the complete content.

Typically called after some user actions in a property widget require that all the other content must be refreshed.

QList<std::shared\_ptr<sh::filesystem::FilesystemNode>> **nodes** ()

The nodes to show.

FilePropertyDialogTabActionsView \***widgetAt** (int *i*)

Returns the widget for the *i*-th property.

int **widgetCount** ()

Returns the number of widgets.

std::shared\_ptr<FilePropertyDialog> **dialog** ()

Returns the associated dialog.

## Public Members

std::function< QList< QString >int, sh::filesystem::Operation \*, qintptr>> **\_updateV**

std::function<void (QString)> **\_buttonTriggered**

### Public Static Attributes

```
const int PROPERTY_TYPE_STRING = 1
const int PROPERTY_TYPE_STRINGMAP = 2
const int PROPERTY_TYPE_ICONTEXTBANNER = 3
```

### Private Types

```
typedef QPair<QString, QString> StringPair
```

### Private Members

```
QString _title
QList<PropertyConfig> _properties
class PropertyConfig
    #include <apifilepropertydialogtab.h>
```

### Public Types

```
typedef QPair<QString, QString> ButtonConfig
```

### Public Functions

```
PropertyConfig (QString title, int propertytype, QList<ButtonConfig> buttons)
```

### Public Members

```
QString title
int propertytype
QList<ButtonConfig> buttons
class ApiFilePropertyDialogTabFactory : public sh::ui::FilePropertyDialogTabFactory
    #include <apifilepropertydialogtab.h>
```

### Public Functions

```
ApiFilePropertyDialogTabFactory ()
std::shared_ptr<sh::ui::FilePropertyDialogTab> construct (std::shared_ptr<sh::ui::FilePropertyDialog>
                                                         dialog)
std::shared_ptr<sh::ui::FilePropertyDialogTab> construct (std::shared_ptr<sh::ui::FilePropertyDialog>
                                                         dialog) = 0
    Constructs a fresh instance of a FilePropertyDialogTab implementation. .
```

## Public Members

```
std::function< std::shared_ptr< sh::scripting::api::ApiFilePropertyDialogTab >> > _c
```

## Public Static Attributes

```
const int REGISTER_FILEPROPERTYDIALOGTAB_INDEX_CORE = 10000
```

Base value for display indexes of core (i.e. maximum important) tabs.

Used in *FilePropertyDialog::addTabFactory*.

```
const int REGISTER_FILEPROPERTYDIALOGTAB_INDEX_VERYIMPORTANT = 20000
```

Base value for display indexes of very important tabs.

Used in *FilePropertyDialog::addTabFactory*.

```
const int REGISTER_FILEPROPERTYDIALOGTAB_INDEX_IMPORTANT = 30000
```

Base value for display indexes of important tabs.

Used in *FilePropertyDialog::addTabFactory*.

```
const int REGISTER_FILEPROPERTYDIALOGTAB_INDEX_NORMAL = 40000
```

Base value for display indexes of tabs with medium importance.

Used in *FilePropertyDialog::addTabFactory*.

```
const int REGISTER_FILEPROPERTYDIALOGTAB_INDEX_EXOTIC = 50000
```

Base value for display indexes of exotic (i.e. less important) tabs.

Used in *FilePropertyDialog::addTabFactory*.

```
class ApiFilesystemHandler : public sh::filesystem::FilesystemHandler
#include <apifilesystemhandler.h>
```

## Public Functions

```
ApiFilesystemHandler (sh::filesystem::FilesystemModel *model)
```

```
void itemList (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
               sh::filesystem::FilesystemNodeType type, sh::filesystem::FilesystemNodeListEditor
               *list)
```

```
void configureItems (sh::filesystem::Operation *op, QList<std::shared_ptr<sh::filesystem::FilesystemNode>>
                     nodes)
```

```
sh::filesystem::FilesystemNodeType getType (sh::filesystem::Operation *op,
                                             std::shared_ptr<const sh::filesystem::Eurl> eurl)
```

```
qint64 getSize (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl)
```

```
QDateTime getMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                    eurl)
```

```
void setMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
               QDateTime time)
```

```
QString getMimeType (sh::filesystem::Operation *op, std::shared_ptr<const
                     sh::filesystem::Eurl> eurl)
```

```
QString getLinkTarget (sh::filesystem::Operation *op, std::shared_ptr<const
                       sh::filesystem::Eurl> eurl)
```



```

QList<QString> listExtendedAttributes (sh::filesystem::Operation *op,
                                         std::shared_ptr<const sh::filesystem::Eurl>
                                         eurl)

quint64 getExtendedAttributeSize (sh::filesystem::Operation *op, std::shared_ptr<const
                                     sh::filesystem::Eurl> eurl, QString attribute)

QByteArray getExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
                                   sh::filesystem::Eurl> eurl, QString attribute)

void setExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
                             sh::filesystem::Eurl> eurl, QString attribute, QByteArray
                             value)

void removeExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
                                sh::filesystem::Eurl> eurl, QString attribute)

QMap<QString, QString> getCustomAttributes (sh::filesystem::Operation
                                                *op, std::shared_ptr<const
                                                sh::filesystem::Eurl> eurl)

void setCustomAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
                           sh::filesystem::Eurl> eurl, QString attribute, QString value)

bool canGetFileContent (sh::filesystem::Operation *op, std::shared_ptr<const
                          sh::filesystem::Eurl> eurl)

std::shared_ptr<QIODevice> getFileContent (sh::filesystem::Operation *op,
                                             std::shared_ptr<const sh::filesystem::Eurl>
                                             eurl)

bool canCreateDirectory (sh::filesystem::Operation *op, std::shared_ptr<const
                           sh::filesystem::Eurl> eurl)

void createDirectory (sh::filesystem::Operation *op, std::shared_ptr<const
                       sh::filesystem::Eurl> eurl)

bool canCreateLink (sh::filesystem::Operation *op, std::shared_ptr<const
                      sh::filesystem::Eurl> eurl)

void createLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                  eurl, QString target)

bool canCreateFile (sh::filesystem::Operation *op, std::shared_ptr<const
                     sh::filesystem::Eurl> eurl)

void createFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                  eurl, QIODevice *content, HandlerTransfer *handlertransfer)

bool canDeleteItem (sh::filesystem::Operation *op, std::shared_ptr<const
                     sh::filesystem::Eurl> eurl)

void deleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                  eurl)

bool canRenameItem (sh::filesystem::Operation *op, std::shared_ptr<const
                     sh::filesystem::Eurl> src)

void renameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> src,
                  QString destpath)

QList<std::shared_ptr<sh::actions::AbstractActionItem>> getActions (const
                                                                    QList<std::shared_ptr<const
                                                                    sh::filesystem::Eurl>> eu-
                                                                    rls)

```

void **itemlist** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, *sh::filesystem::FilesystemNodeType* type, *sh::filesystem::FilesystemNodeListEditor* \*list) = 0

Determine a list of subelements in a certain directory.

void **configureItems** (*sh::filesystem::Operation* \*op, QList<std::shared\_ptr<FilesystemNode>> nodes)

Configure newly created nodes (e.g. setting another icon or changing the display name).

*sh::filesystem::FilesystemNodeType* **getType** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0

Determine if a file is regular, or a dir, or a link, ...

qint64 **getSize** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0

Determine the size of a file.

QDateTime **getMtime** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0

Determine the mtime of a file.

void **setMtime** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QDateTime time) = 0

Set the mtime of a file.

QString **getMimetype** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0

Determine mime type of a file.

QString **getLinkTarget** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0

Resolve a link.

QList<QString> **listExtendedAttributes** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)

Fetches the list of available extended attributes for an entry.

qint64 **getExtendedAttributeSize** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QString attribute)

Returns the size of value for one extended attribute for an entry.

QByteArray **getExtendedAttribute** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QString attribute)

Returns the value for one extended attribute for an entry.

void **setExtendedAttribute** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QString attribute, QByteArray value)

Sets the value for one extended attribute for an entry.

void **removeExtendedAttribute** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QString attribute)

Removes one extended attributes for an entry.

QMap<QString, QString> **getCustomAttributes** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)

Returns the custom attributes for an entry.

In contrast to extended attributes, the custom attributes are not directly stored in the filesystem this way, but handle implementation specific aspects (like permissions).

void **setCustomAttribute** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QString attribute, QString value)  
 Sets the value for one custom attribute for an entry.  
 See also getCustomAttributes.

bool **canGetFileContent** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0  
 Returns if it is allowed to get the content of a certain file.

std::shared\_ptr<QIODevice> **getFileContent** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0  
 Get the content of a file.

bool **canCreateDirectory** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0  
 Returns if it is allowed to create subdirectories in a certain directory.

void **createDirectory** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0  
 Create a directory.

bool **canCreateLink** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0  
 Returns if it is allowed to create a link in a certain directory.

void **createLink** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QString target) = 0  
 Create a link.

bool **canCreateFile** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0  
 Returns if it is allowed to create files in a certain directory.

void **createFile** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QIODevice \*content, HandlerTransfer \*handlertransfer) = 0  
 Creates a file with some content.  
 It may use handlertransfer for some better integration, like cancel support and progress monitoring.

bool **canDeleteItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0  
 Returns if it is allowed to delete a certain file/directory/link/...

void **deleteItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0  
 Delete a file/directory/link/...

void **deleteDirectoryIfEmpty** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)  
 Deletes a directory only if it is empty.

bool **canRenameItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> src) = 0  
 Returns if it is allowed to rename a certain file, directory, link, ... (see renameItem).

void **renameItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> src, QString destpath) = 0  
 Renames an item to another path.  
 It can be a simple filename change or also changes in the deeper path segments.

`QList<std::shared_ptr<sh::actions::AbstractActionItem>> getActions (const`  
`QList<std::shared_ptr<const`  
`sh::filesystem::Eurl>> eu-`  
`rls) = 0`  
Returns a list of actions (see former example) for showing for certain files.

`bool requiresResolvedLinks ()`  
Returns if this handler requires to be used by the engine with resolved links.

`void viewEnteredDirectory (std::shared_ptr<const sh::filesystem::Eurl>)`  
Callback for preparing stuff when the view entered some directory (e.g. for enabling watchers). See also `viewLeftDirectory`.

`void viewLeftDirectory (std::shared_ptr<const sh::filesystem::Eurl>)`  
Callback for preparing stuff when the view left some directory. See also `viewEnteredDirectory`.

`bool isWellKnownScheme ()`  
Returns if the scheme for this handler is a well known one (which external programs typically would understand).

`void search (sh::filesystem::Operation *op, std::shared_ptr<const`  
`sh::filesystem::Eurl> eurl, std::function<void> std::shared_ptr<const`  
`sh::filesystem::Eurl>, QList<std::shared_ptr<sh::search::SearchCriterion>>,`  
`sh::filesystem::FilesystemNodeType ftype`  
`> addfile, std::function<void>std::shared_ptr<const sh::filesystem::Eurl>> visitdir,`  
`QList<std::shared_ptr<sh::search::SearchCriterion>> criteria`Helps searching for files.

Override this method with a behavior identical to the default, if a specialized implementation can be much faster than the default one.

`void customizeUi (std::shared_ptr<sh::filesystem::FilesystemNode> node, bool *showSearch-`  
`Panel)`  
Creates a custom gui, which becomes part of the fileview.

`sh::filesystem::FilesystemModel *model ()`  
A convenience shortcut to the `sh::filesystem::FilesystemModel` (a singleton).

## Public Members

`std::function<int (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl) >`  
`_getType`

`std::function<qint64 (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl) >`  
`_getSize`

`std::function<double (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl) >`  
`_getMtime`

`std::function<void (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl, dou-`  
`ble) > _setMtime`

`std::function<QString (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl) >`  
`_getLinkTarget`

`std::function< QList< QString >sh::filesystem::Operation *op, std::shared_ptr< sh::`

`std::function<quint64 (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl,`  
`QString attribute) > _getExtendedAttributeSize`

`std::function<QByteArray (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl,`  
`QString attribute) > _getExtendedAttribute`

`std::function<void (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl, QString`  
`attribute, QByteArray value) > _setExtendedAttribute`

```

std::function<void (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl, QString
attribute) > _removeExtendedAttribute

std::function< QList< QString > sh::filesystem::Operation *op, std::shared_ptr< sh::
std::function<void (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl, QString
attribute, QString value) > _setCustomAttribute

std::function<QString (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl) >
_getMimeType

std::function<bool (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl) >
_canCreateDirectory

std::function<void (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl) >
_createDirectory

std::function<bool (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl) >
_canCreateLink

std::function<void (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl, QString
target) > _createLink

std::function<bool (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl) >
_canCreateFile

std::function<bool (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl) >
_canDeleteItem

std::function<void (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl) >
_deleteItem

std::function<bool (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> src) >
_canRenameItem

std::function<void (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> src, QString
destpath) > _renameItem

std::function<void (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl, int type,
sh::filesystem::FilesystemNodeListEditor *list) > _itemlist

std::function<void (sh::filesystem::Operation *op, QList<std::shared_ptr<sh::filesystem::FilesystemNode>>) >
_configureItems

std::function<bool (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl) >
_canGetFileContent

std::function< std::shared_ptr< QIODevice > sh::filesystem::Operation *op, std::sha
std::function<void (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl, QIODe-
vice *content, HandlerTransfer *handlertransfer) > _createFile

std::function< QList< std::shared_ptr< sh::actions::AbstractActionItem > >QList< st

class ApiFilesystemOperationProgressMonitor : public sh::filesystem::FilesystemOperationProgressMonitor
#include <apifilesystemoperationprogressmonitor.h>

```

## Public Functions

**ApiFilesystemOperationProgressMonitor** (*sh::actions::ActionExecutionInfo* \*actionExecution)

bool **hasItemInfo** ()  
Checks if this progress monitor provides information about how many items of a certain total number are transferred so far.

quint64 **doneItems** ()  
Checks how many items are transferred so far.

quint64 **allItems** ()  
Checks how many items are to be transferred in total (as predicted in the current moment).

bool **hasBytesInfo** ()  
Checks if this progress monitor provides information about how many byte of a certain total number are transferred so far.

quint64 **doneBytes** ()  
Checks how many bytes are transferred so far.

quint64 **allBytes** ()  
Checks how many bytes are to be transferred in total (as predicted in the current moment).

QString **getItemInfoFrom** ()  
Returns the current source of transfer (as textual information).

QString **getItemInfoTo** ()  
Returns the current destination of transfer (as textual information).

QString **estimation** ()  
Returns the current performance and time estimation (as textual information).

## Public Members

std::function<void ()> **\_changed**

std::function<bool (QList<sh::filesystem::FilesystemOperationTransfers::OperationStep\*>)>  
**\_resolveConflicts**

**class ApiGlobalObject : public sh::base::Singleton**  
*#include <apiglobalobject.h>*

## Public Functions

std::shared\_ptr<const sh::filesystem::Eurl> **getEurlFromString** (QString eurl)

void **logDebug** (QString msg)

void **logInfo** (QString msg)

void **logWarning** (QString msg)

void **logError** (QString msg)

int **filesystemnodetype\_file** ()

int **filesystemnodetype\_directory** ()

int **filesystemnodetype\_link** ()

```

int filesystemnodetype_unknown()
int filesystemnodetype_firsttype()
int filesystemnodetype_lastphysicaltype()
int filesystemnodetype_none()
int filesystemnodetype_specialtreeonlydirectory()
int filesystemnodetype_lasttype()
int actiondefaultprecedencevalues_openfile()
int actiondefaultprecedencevalues_builtinspecialopen()
int messageboxbutton_ok()
int messageboxbutton_continue()
int messageboxbutton_cancel()
int messageboxbutton_retry()
int messageboxbutton_yes()
int messageboxbutton_no()
int displayindex_core()
int displayindex_interesting()
int displayindex_exotic()
int configurationcategory_gui()
int configurationcategory_behavior()
int configurationcategory_externaltools()
int settinggroup_global()
int settinggroup_gui()
int settinggroup_filehandling()
int settinggroup_dataview()
int settinggroup_behavior()
int settinggroup_special()
int paneldetail_positionindex_veryinteresting()
int paneldetail_positionindex_interesting()
int paneldetail_positionindex_exotic()
int operationstep_conflictresolution_mergeddirectories()
int operationstep_conflictresolution_overwritedestination()
int operationstep_conflictresolution_renamedestinationbefore()
int operationstep_conflictresolution_skip()
int operationstep_conflictresolution_unresolved()
int operationstep_conflictresolution_indirect()
int operationstep_conflictresolution_usedifferentdestinationname()

```

```
int filepropertydialogtab_index_core ()
int filepropertydialogtab_index_veryimportant ()
int filepropertydialogtab_index_important ()
int filepropertydialogtab_index_normal ()
int filepropertydialogtab_index_exotic ()
int filepropertydialogtab_propertytype_string ()
int filepropertydialogtab_propertytype_stringmap ()
int filepropertydialogtab_propertytype_icontextbanner ()
int searchcriterionfactory_index_core ()
int searchcriterionfactory_index_normal ()
int searchcriterionfactory_index_exotic ()
int thumbnailprovider_index_core ()
int thumbnailprovider_index_normal ()
int thumbnailprovider_index_exotic ()
int thumbnailprovider_index_fallback ()
void register_predicatedactionfactory (std::shared_ptr<sh::scripting::api::ApiActionFactory>
                                     f)
bool isDebugBuild ()
std::shared_ptr<sh::scripting::api::ApiFilesystemHandler> create_FilesystemHandler ()
std::shared_ptr<sh::scripting::api::ApiActionActionItem> create_ActionActionItem (QString
                                     text,
                                     bool
                                     en-
                                     abled,
                                     QString
                                     icon,
                                     int de-
                                     fault-
                                     Ac-
                                     tion-
                                     Prece-
                                     dence)
```



```

std::shared_ptr<sh::scripting::api::ApiSubmenuItem> create_SubmenuItem (QString
                                                                    text,
                                                                    bool
                                                                    en-
                                                                    abled,
                                                                    QString
                                                                    icon,
                                                                    int
                                                                    de-
                                                                    fault-
                                                                    Ac-
                                                                    tion-
                                                                    Prece-
                                                                    dence)

std::shared_ptr<sh::scripting::api::ApiDetailColumn> create_DetailColumn (QString name,
                                                                    QString display-
                                                                    Name, int displayIndex, bool
                                                                    sort_doTypediff,
                                                                    int default-
                                                                    Width, bool
                                                                    isRightAligned)

std::shared_ptr<sh::scripting::api::ApiFilePropertyDialogTab> create_FilePropertyDialogTab (QString
                                                                    ti-
                                                                    tle,
                                                                    QList<QString>
                                                                    prop-
                                                                    er-
                                                                    ties)

std::shared_ptr<sh::scripting::api::ApiFilePropertyDialogTabFactory> create_FilePropertyDialogTabFactory

std::shared_ptr<sh::scripting::api::ApiActionFactory> create_ActionFactory (QString
                                                                    category,
                                                                    QList<std::shared_ptr<actions::Predi-
                                                                    cates>
                                                                    predicates)

std::shared_ptr<sh::scripting::api::ApiPanelDetailFactorySingle> create_DetailFactorySingle (int
                                                                    po-
                                                                    si-
                                                                    tion,
                                                                    int
                                                                    val-
                                                                    ueWidth-
                                                                    Hint)

std::shared_ptr<sh::scripting::api::ApiPanelDetailFactoryMulti> create_DetailFactoryMulti (int
                                                                    po-
                                                                    si-
                                                                    tion,
                                                                    int
                                                                    val-
                                                                    ueWidth-
                                                                    Hint)

```

```
std::shared_ptr<sh::scripting::api::ApiFilesystemOperationProgressMonitor> create_FilesystemOperationProgre

std::shared_ptr<sh::filesystem::FilesystemNode> _createFilesystemNode (std::shared_ptr<sh::filesystem::Eurl>
                                                                    eurl,
                                                                    sh::scripting::api::ApiFilesystemHandler
                                                                    *handler,
                                                                    int      nodetype,
                                                                    std::shared_ptr<sh::filesystem::Filesystem
                                                                    parent, bool doin-
                                                                    sert, bool showIni-
                                                                    tialLoadingLabel,
                                                                    bool isHidden)

std::shared_ptr<sh::scripting::api::ApiThumbnailProvider> create_ThumbnailProvider ()

std::shared_ptr<sh::filesystem::FilesystemNode> _getOrCreateFilesystemNode (std::shared_ptr<sh::filesystem::Eurl>
                                                                    eurl,
                                                                    sh::scripting::api::ApiFilesystemH
                                                                    *han-
                                                                    dler,      int
                                                                    nodetype,
                                                                    std::shared_ptr<sh::filesystem::File
                                                                    parent,
                                                                    bool doin-
                                                                    sert,      bool
                                                                    showIni-
                                                                    tialLoad-
                                                                    ingLabel,
                                                                    bool isHid-
                                                                    den)

void _register_FilesystemHandler (std::shared_ptr<sh::scripting::api::ApiFilesystemHandler>
                                handler, QString scheme)

std::shared_ptr<sh::filesystem::FilesystemNode> modelRootNode ()

QList<std::shared_ptr<sh::filesystem::FilesystemNode>> _findFilesystemNodesForEurl (std::shared_ptr<const
                                                                    sh::filesystem::Eurl>
                                                                    eurl)

QList<std::shared_ptr<sh::filesystem::FilesystemNode>> _tryGetFilesystemNodesForEurl (std::shared_ptr<const
                                                                    sh::filesystem::Eurl>
                                                                    eurl)

void refreshData (std::shared_ptr<const sh::filesystem::Eurl> eurl, bool forceFindParent, bool
                  withDetails)

std::shared_ptr<sh::filesystem::Eurl> createEurl (QString scheme, QString hostname, QString path)

std::shared_ptr<sh::filesystem::Operation> createOperation ()

sh::scripting::StringMap<QString> createArgumentException ()

sh::scripting::StringMap<QString> createIOException ()

sh::scripting::StringMap<QString> createRuntimeException ()

sh::scripting::StringMap<QString> createProgramException ()
```

```

sh::scripting::StringMap<QString> createException ()

std::shared_ptr<sh::scripting::api::ApiReadDataDevice> _createReadDataDevice ()

std::shared_ptr<ApiSetting> createSetting (QString name, QString description, int group, bool
                                         isAdvancedSetting, bool isGlobal, bool isPerFile-
                                         view)

std::shared_ptr<sh::scripting::api::ApiThread> create_Thread ()

std::shared_ptr<sh::scripting::api::ApiTimer> create_Timer ()

std::shared_ptr<sh::scripting::api::ApiSearchCriterion> create_SearchCriterion (std::shared_ptr<sh::scripting::api::ApiSearchCriterionFactory>
                                         factory)

std::shared_ptr<sh::scripting::api::ApiSearchCriterionFactory> create_SearchCriterionFactory (QString
                                                                 key,
                                                                 QString
                                                                 de-
                                                                 scrip-
                                                                 tion)

sh::ui::MainWindow *mainwindow ()

void registerPanelDetailFactorySingle (std::shared_ptr<sh::scripting::api::ApiPanelDetailFactorySingle>)

void registerPanelDetailFactoryMulti (std::shared_ptr<sh::scripting::api::ApiPanelDetailFactoryMulti>)

void registerSetting (std::shared_ptr<sh::settings::Setting> setting)

void openItems (QList<std::shared_ptr<const sh::filesystem::Eurl>> eurls)

void registerFilePropertyDialogTabFactory (int index,
                                             std::shared_ptr<sh::scripting::api::ApiFilePropertyDialogTabFactory>
                                             factory)

void registerThumbnailProvider (int idx, std::shared_ptr<sh::scripting::api::ApiThumbnailProvider>
                                thumbnailprovider)

void registerSearchCriterionFactory (int idx, std::shared_ptr<ApiSearchCriterionFactory>
                                     apicrit)

std::shared_ptr<sh::configuration::ConfigurationValue> register_ConfigurationValueInt (QString
                                                                 name,
                                                                 int
                                                                 de-
                                                                 flt,
                                                                 QString
                                                                 desc,
                                                                 int
                                                                 cat-
                                                                 e-
                                                                 gory,
                                                                 QString
                                                                 longdesc,
                                                                 QString
                                                                 change-
                                                                 hint)

```

```
std::shared_ptr<sh::configuration::ConfigurationValue> register_ConfigurationValueFloat (QString
                                                    name,
                                                    int
                                                    de-
                                                    flt,
                                                    QString
                                                    desc,
                                                    int
                                                    cat-
                                                    e-
                                                    gory,
                                                    QString
                                                    longdesc,
                                                    QString
                                                    change-
                                                    hint)

std::shared_ptr<sh::configuration::ConfigurationValue> register_ConfigurationValueBool (QString
                                                    name,
                                                    bool
                                                    de-
                                                    flt,
                                                    QString
                                                    desc,
                                                    int
                                                    cat-
                                                    e-
                                                    gory,
                                                    QString
                                                    longdesc,
                                                    QString
                                                    change-
                                                    hint)

std::shared_ptr<sh::configuration::ConfigurationValue> register_ConfigurationValueString (QString
                                                    name,
                                                    QString
                                                    de-
                                                    flt,
                                                    QString
                                                    desc,
                                                    int
                                                    cat-
                                                    e-
                                                    gory,
                                                    QString
                                                    longdesc,
                                                    QString
                                                    change-
                                                    hint)

void registerTransferrableDetailColumn (int i, std::shared_ptr<sh::filesystem::DetailColumn>
                                         detailColumn)

std::shared_ptr<sh::filesystem::DetailColumn> findDetailColumnByName (QString name)

QString shallotDataDir ()
```

```

QString shallotBuiltinPluginDir ()
QString shallotSystemPluginDir ()
QString shallotUserPluginDir ()
QString shallotLanguage ()
sh::tools::BookmarkManager *bookmarkManager ()
std::shared_ptr<sh::actions::Predicate> create_OnDirectoriesPredicate ()
std::shared_ptr<sh::actions::Predicate> create_OnFilesPredicate ()
std::shared_ptr<sh::actions::Predicate> create_OnLinksPredicate ()
std::shared_ptr<sh::actions::Predicate> create_OnSingleEntrySelectionPredicate ()
std::shared_ptr<sh::actions::Predicate> create_DontResolveLinksPredicate ()
std::shared_ptr<sh::actions::Predicate> create_HideOnCurrentDirectoryLevelPredicate ()
std::shared_ptr<sh::actions::Predicate> create_HideOnSelectionLevelPredicate ()
std::shared_ptr<sh::actions::Predicate> create_ByRegExpPredicate (QString pattern)
std::shared_ptr<sh::actions::Predicate> create_KeyShortcutPredicate (QString shortcut,
                                                                    bool triggersOn-
                                                                    CurrentDirecto-
                                                                    ryLevel)
std::shared_ptr<sh::actions::Predicate> create_PositionIndexPredicate (int index)
void doInitialize ()
    Executes singleton initialization.
void doShutdown ()
    Executes singleton shutdown.
void shutdown ()
    Shutdown down this singleton.
bool isAlive ()
    Returns if this singleton is alive (true until its shutdown begins).

```

## Private Functions

```

ApiGlobalObject ()

```

```

class ApiHelperMethods
    #include <apihelpermethods.h>

```

## Public Static Functions

```

QByteArray QIODevice_read (QIODevice *self)
QByteArray QIODevice_readall (QIODevice *self)
QByteArray ApiReadDataDevice_read (sh::scripting::api::ApiReadDataDevice *self)
QByteArray ApiReadDataDevice_readall (sh::scripting::api::ApiReadDataDevice *self)
void FilesystemNodeList_resetItems (sh::filesystem::FilesystemNodeList *self, int type,
                                     QList<std::shared_ptr<sh::filesystem::FilesystemNode>>
                                     list)

```

```
void FilesystemNode_addDetail (filesystem::FilesystemNode *self,
                               std::shared_ptr<sh::filesystem::DetailColumn> col)
void FilesystemNode_setIcon (sh::filesystem::FilesystemNode *self, QString icon)
void FilesystemNode_setDisplayName (filesystem::FilesystemNode *self, QString display-
                                     name)
bool FilesystemNode_isHidden (filesystem::FilesystemNode *self)
void FilesystemNode_setHidden (filesystem::FilesystemNode *self, bool v)
void ConfigurationValue_setConfigValueInt (sh::configuration::ConfigurationValue
                                             *self, int v)
void ConfigurationValue_setConfigValueFloat (sh::configuration::ConfigurationValue
                                                *self, double v)
void ConfigurationValue_setConfigValueBool (sh::configuration::ConfigurationValue
                                               *self, bool v)
void ConfigurationValue_setConfigValueString (sh::configuration::ConfigurationValue
                                                *self, QString v)
QList<std::shared_ptr<sh::filesystem::Eurl>> FilesystemOperation_itemlist (sh::filesystem::FilesystemOperation
                                                                            *self,
                                                                            std::shared_ptr<const
                                                                            sh::filesystem::Eurl>
                                                                            eurl)
QList<std::shared_ptr<sh::filesystem::Eurl>> FilesystemOperation_itemlistByType (sh::filesystem::FilesystemOperation
                                                                                  *self,
                                                                                  std::shared_ptr<const
                                                                                  sh::filesystem::Eurl>
                                                                                  eurl,
                                                                                  int
                                                                                  ntype)
int FilesystemOperation_getType (sh::filesystem::FilesystemOperation *self,
                                  std::shared_ptr<const sh::filesystem::Eurl> eurl)
void FilesystemOperation_createFile (sh::filesystem::FilesystemOperation
                                       *self,
                                       std::shared_ptr<const
                                       sh::filesystem::Eurl>
                                       eurl,
                                       sh::scripting::api::ApiReadDataDevice *content,
                                       std::shared_ptr<filesystem::FilesystemOperationProgressMonitor>
                                       progressmon)
QList<std::shared_ptr<sh::filesystem::FilesystemNode>> FilesystemOperation_resolveNodeLink (sh::filesystem::FilesystemNode
                                                                                          *self,
                                                                                          std::shared_ptr<const
                                                                                          sh::filesystem::FilesystemNode>
                                                                                          node)
QList<std::shared_ptr<sh::filesystem::FilesystemNode>> FilesystemOperation_resolveNodeLink_NonRecursive (sh::filesystem::FilesystemNode
                                                                                                      *self,
                                                                                                      std::shared_ptr<const
                                                                                                      sh::filesystem::FilesystemNode>
                                                                                                      node)
std::shared_ptr<sh::filesystem::Eurl> FilesystemOperation_resolveEurlLink (sh::filesystem::FilesystemOperation
                                                                              *self,
                                                                              std::shared_ptr<const
                                                                              sh::filesystem::Eurl>
                                                                              eurl)
```

```

std::shared_ptr<sh::filesystem::Eurl> FilesystemOperation_resolveEurlLink_NonRecursive (sh::filesystem
                                                                    *self,
                                                                    std::shared_p
                                                                    sh::filesystem
                                                                    eurl)

void PanelDetail_setRow (sh::paneldetails::PanelDetail *self, QString key, QList<QString>
                                                                    value)

void MainWindow_jumpToEurl (sh::ui::MainWindow *self, std::shared_ptr<const
                                                                    sh::filesystem::Eurl> eurl)

std::shared_ptr<sh::filesystem::FilesystemNode> MainWindow_getCurrentDirectoryNode (sh::ui::MainWindow
                                                                    *self)

void ApiActionActionItem_setEnabled (sh::scripting::api::ApiActionActionItem *self,
                                                                    bool enabled)

bool ApiActionActionItem_enabled (sh::scripting::api::ApiActionActionItem *self)

void ApiActionActionItem_setVisible (sh::scripting::api::ApiActionActionItem *self,
                                                                    bool visible)

bool ApiActionActionItem_visible (sh::scripting::api::ApiActionActionItem *self)

void ApiSubmenuItem_setSubitems (sh::scripting::api::ApiSubmenuItem
                                                                    *self, QList<std::shared_ptr<actions::AbstractActionItem>>
                                                                    subitems)

void ApiSubmenuItem_setEnabled (sh::scripting::api::ApiSubmenuItem *self,
                                                                    bool enabled)

bool ApiSubmenuItem_enabled (sh::scripting::api::ApiSubmenuItem *self)

void ApiSubmenuItem_setVisible (sh::scripting::api::ApiSubmenuItem *self,
                                                                    bool visible)

bool ApiSubmenuItem_visible (sh::scripting::api::ApiSubmenuItem *self)

int OperationStep_conflictResolution (filesystem::FilesystemOperationTransfers::OperationStep
                                                                    *self)

bool FilesystemOperationProgressMonitor_hasItemInfo (sh::scripting::api::ApiFilesystemOperationProgr
                                                                    *self)

quint64 FilesystemOperationProgressMonitor_doneItems (sh::scripting::api::ApiFilesystemOperationProgr
                                                                    *self)

quint64 FilesystemOperationProgressMonitor_allItems (sh::scripting::api::ApiFilesystemOperationProgre
                                                                    *self)

bool FilesystemOperationProgressMonitor_hasBytesInfo (sh::scripting::api::ApiFilesystemOperationProgr
                                                                    *self)

quint64 FilesystemOperationProgressMonitor_doneBytes (sh::scripting::api::ApiFilesystemOperationProgr
                                                                    *self)

quint64 FilesystemOperationProgressMonitor_allBytes (sh::scripting::api::ApiFilesystemOperationProgre
                                                                    *self)

QString FilesystemOperationProgressMonitor_getItemInfoFrom (sh::scripting::api::ApiFilesystemOpera
                                                                    *self)

QString FilesystemOperationProgressMonitor_getItemInfoTo (sh::scripting::api::ApiFilesystemOperation
                                                                    *self)

QString FilesystemOperationProgressMonitor_estimation (sh::scripting::api::ApiFilesystemOperationPro
                                                                    *self)

```

```
QList<std::shared_ptr<sh::filesystem::FilesystemNode>> ApiFilePropertyDialogTab_nodes (sh::scripting::api::A
                                                                    *self)

int ActionExecutionUserFeedback_messageBox (sh::actions::ActionExecutionUserFeedback
                                                                    *self, QString m, QList<QString> l,
                                                                    QString s, int i, int j)

QString ActionExecutionUserFeedback_simpleInputDialog (sh::actions::ActionExecutionUserFeedback
                                                                    *self, QString text, QString
                                                                    deflt)

int FilePropertyDialogTab_selectedIndexForProperty (ApiFilePropertyDialogTab
                                                                    *self, qintptr widget)

void FilePropertyDialogTab_refresh (ApiFilePropertyDialogTab *self)

class ApiPanelDetailFactoryMulti : public sh::paneldetails::PanelDetailFactoryForFunctionMulti
#include <apipaneldetailfactory.h>
```

### Public Functions

```
ApiPanelDetailFactoryMulti (int position, int valueWidthHint)

void setvalue (std::shared_ptr<sh::paneldetails::PanelDetail> detail,
               QList<std::shared_ptr<sh::filesystem::FilesystemNode>> nodes, sh::filesystem::Operation *op)

void linktriggered (QList<std::shared_ptr<sh::filesystem::FilesystemNode>> nodes, QString linktarget)

std::shared_ptr<PanelDetail> construct (QList<std::shared_ptr<sh::filesystem::FilesystemNode>>
                                       nodes, sh::filesystem::Operation *op)
```

### Public Members

```
std::function<void (sh::paneldetails::PanelDetail*, QList<std::shared_ptr<sh::filesystem::FilesystemNode>>,
                   sh::filesystem::Operation *op)> _setvalue

std::function<void (QList<std::shared_ptr<sh::filesystem::FilesystemNode>>, QString) >
_linktriggered

class ApiPanelDetailFactorySingle : public sh::paneldetails::PanelDetailFactoryForFunctionSingle
#include <apipaneldetailfactory.h>
```

### Public Functions

```
ApiPanelDetailFactorySingle (int position, int valueWidthHint)

void setvalue (std::shared_ptr<sh::paneldetails::PanelDetail>, std::shared_ptr<sh::filesystem::FilesystemNode>
               node, sh::filesystem::Operation *op)

void linktriggered (std::shared_ptr<sh::filesystem::FilesystemNode> node, QString linktarget)

std::shared_ptr<PanelDetail> construct (QList<std::shared_ptr<sh::filesystem::FilesystemNode>>
                                       nodes, sh::filesystem::Operation *op)
```



## Public Members

```
std::function<void (sh::paneldetails::PanelDetail*, std::shared_ptr<sh::filesystem::FileSystemNode>,
                  sh::filesystem::Operation *op)> _setvalue
std::function<void (std::shared_ptr<sh::filesystem::FileSystemNode>,                      QString)>
                  _linktriggered
class ApiReadDataDevice : public sh::tools::ReadDataDevice
#include <apireaddatadevice.h>
```

## Public Functions

**ApiReadDataDevice** ()

QByteArray **getdata** ()

This method returns the next available chunk of data. It may return empty arrays whenever temporarily no data is available. Returning a null array (with `QByteArray::isNull() == true`) marks the end of the stream. .

void **\_ended** ()

## Public Members

std::function<QByteArray ()> **\_getdata**

## Private Members

bool **isended**

```
class ApiSearchCriterion : public sh::search::criteria::AbstractActionDrivenSearchCriterion
#include <apisearchcriterion.h>
```

## Public Functions

**ApiSearchCriterion** (std::shared\_ptr<sh::search::SearchCriterionFactory> factory)

bool **match2** (sh::filesystem::Operation \*op, std::shared\_ptr<const sh::filesystem::Eurl> eurl)

QList<QString> **getsearchspec** ()

bool **match** (sh::filesystem::Operation \*op, std::shared\_ptr<const sh::filesystem::Eurl> eurl)  
Checks if a given file (by eurl) matches this criterion. .

QString **valuedescription** ()

Returns a human readable text describing the current criterion shortly and precisely. .

QString **serialize** ()

Serializes this criterion to a string. See also `SearchCriterion::deserialize`.

std::shared\_ptr<sh::search::SearchCriterionFactory> **factory** ()

Returns the factory this criterion has created (provides some more metadata).

## Public Members

```
std::function<void (sh::actions::ActionExecutionInfo*)> _configure
std::function<bool (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::Eurl> eurl)>
_match
QStringList searchspec
```

## Public Static Functions

```
void _createEditor (std::shared_ptr<sh::search::SearchCriterion>                                cfg,
                    sh::ui::SearchPanelConfiguration *panelcfg)
void _editorUpdateConfiguration (sh::ui::SearchPanelConfiguration *panelcfg,
                                std::shared_ptr<sh::search::SearchCriterion> c)
std::shared_ptr<sh::search::SearchCriterion> deserialize (QString s)
    Deserializes this criterion from a string. See also SearchCriterion::serialize.
class ApiSearchCriterionFactory : public sh::search::SearchCriterionFactory
    #include <apisearchcriterion.h>
```

## Public Functions

```
ApiSearchCriterionFactory (QString key, QString description)
QString key ()
    Returns the string key for the associated search criterion class. .
QString description ()
    Returns the description string for the associated search criterion class. .
void editor (std::shared_ptr<sh::search::SearchCriterion> c, sh::ui::SearchPanelConfiguration
            *panelcfg)
void editorupdateconfig (sh::ui::SearchPanelConfiguration *panelcfg,
                        std::shared_ptr<sh::search::SearchCriterion> c)
std::shared_ptr<sh::search::SearchCriterion> construct ()
    Constructs an instance of the associated search criterion class. .
bool isVisibleFor (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::FilesystemNode>
                  node)
    Checks if the associated search criterion class should be offered to the user for a given node. .
void editor (std::shared_ptr<sh::search::SearchCriterion> c, sh::ui::SearchPanelConfiguration
            *panelcfg) = 0
    Builds the search config ui in a search panel.
    Takes the data from c and populates panelcfg with it.
void editorupdateconfig (sh::ui::SearchPanelConfiguration *panelcfg,
                        std::shared_ptr<sh::search::SearchCriterion> c) = 0
    Updates own data with the content from the search config ui in a search panel.
    Takes the data from panelcfg and updates c with it.
```

## Public Members

```
std::function<QString (QList<QString>)> _getsearchspecdesc
std::function< std::shared_ptr< sh::search::SearchCriterion >> _construct
std::function<bool (sh::filesystem::Operation*, std::shared_ptr<sh::filesystem::FilesystemNode>)>
    _isVisibleFor
class ApiSetting: public sh::settings::Setting
    #include <apisetting.h>
```

## Public Functions

**ApiSetting** (QString *name*, QString *description*, sh::settings::SettingGroup *group*, bool *isAdvancedSetting*, bool *isGlobal*, bool *isPerFileview*)

QString **name** ()  
Gets the internal name.

QString **description** ()  
Gets the description text.

sh::settings::SettingGroup **group** ()  
Gets the group;.

bool **isAdvancedSetting** ()  
Is this an advanced setting?

void **setValue** (QString *value*)  
Called from Shallot core when the value was set (for a not-per-fileview setting).

void **setValue** (sh::ui::FileView \**filelist*, QString *value*)  
Called from Shallot core when the value was set (for a per-fileview setting).

QString **getValue** (sh::ui::FileView \**filelist*)  
Get the currently set value.

bool **isGlobal** ()  
Does this setting apply globally or just for a certain subtree of nodes?

bool **isPerFileview** ()  
Does this setting apply for each fileview individually or for the complete main window?

QString **valueDescription** (QString *value*)  
Gets a human readable description text for a value.

## Public Members

```
std::function<void (QString)> _setValue1
std::function<void (int, QString)> _setValue2
std::function<QString (int)> _getValue
std::function<QString (QString)> _valueDescription
QString _name
QString _description
sh::settings::SettingGroup _group
```

```
bool _isAdvancedSetting  
bool _isGlobal  
bool _isPerFileview
```

## Public Static Functions

QString **getGroupDescription** (int g)  
Low-level function which gets the description text of a group.

```
class ApiSubmenuItem: public sh::actions::SubmenuItem  
    #include <apiaction.h>
```

## Public Functions

**ApiSubmenuItem** (QString *text*, bool *enabled*, QString *icon*, int *defaultActionPrecedence*)

void **initialize** () **override**  
Initialize the action. This should make the time-consuming parts, e.g. for determining a label or enabled state.

const QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> **subitems** ()  
Returns the list of subitems from this submenu.

void **setSubitems** (const QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> *subitems*)  
Sets the subitems.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
This e.g. leads to a bold label.

The action with the highest value becomes the default. Only values >0 will be considered. See [\*ActionDefaultPrecedenceValues\*](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

```

void setChecked (bool checked)
    Sets if the item is checked (has a cross in the ui).

void setVisible (bool visible)
    Sets the visibility of this item.

bool visible ()
    Checks the visibility of this item (non-recursively).

std::weak_ptr<AbstractActionItem> parentAction ()
    Returns the parent action, if it is added to a container.

void __setParentAction (std::shared_ptr<AbstractActionItem> parent)
    Sets the parent action. .

void initializeAsync (std::function<void>
    > oninitialized = 0) Asynchronously initializes the action.

void initializeSync ()
    Synchronously initializes the action.

bool isInitialized ()
    Checks if this action is initialized.

bool isInitializing ()
    Checks if this action is initializing.

```

## Public Members

```
std::function<void ()> __initialize
```

## Signals

```

void subitemsChanged ()
    Emits when the list of subitems changed.

void changed ()
    Emits when some data changed.

```

```

class ApiThread: public std::enable_shared_from_this<ApiThread>
    #include <apithread.h>

```

## Public Functions

```

ApiThread ()

void start ()

```

### Public Members

std::function<void ()> **\_run**

```
class ApiThumbnailProvider : public sh::tools::ThumbnailProvider  
    #include <apithumbnailprovider.h>
```

### Public Functions

**ApiThumbnailProvider**()

void **getThumbnail** (*sh::filesystem::Operation* \*operation, std::shared\_ptr<*sh::filesystem::FileSystemNode*>  
 node, QString contentType, int width, int height, QIcon \*icon)

### Public Members

std::function<QByteArray (*sh::filesystem::Operation*\*, std::shared\_ptr<*sh::filesystem::FileSystemNode*>, QString, int, int)> **\_getThumbnail**

```
class ApiTimer : public std::enable_shared_from_this<ApiTimer>  
    #include <apitimer.h>
```

### Public Functions

**ApiTimer**()

void **start** (int interval)

void **stop** ()

### Public Members

std::function<void ()> **\_run**

### Private Members

QMutex **\_mutex**

QTimer **\_timer**

bool **\_isexecuting** = false

QList<std::shared\_ptr<*ApiTimer*>> **\_runningTimers**

## 10.2.16 Namespace *sh::scripting::pythoninterop*

**namespace** *sh::scripting::pythoninterop*

Interoperation layer between the Shallot core (C++) and Python plugins.

## Typedefs

```
using StringMap = QMap<QString, V>
```

## Functions

```
PyObject *getPyNone ()
bool isPyNone (PyObject *o)
void *getPointerFromShallotPyObject (PyObject *o)
std::shared_ptr<void> getSharedPointerFromShallotPyObject (PyObject *o)
PyObject *getShallotPyObjectFromPointer (void *o, sh::scripting::ApiClass *apiclass)
PyObject *getShallotPyObjectFromSharedPointer (std::shared_ptr<void> o,
sh::scripting::ApiClass *apiclass)
sh::scripting::ApiClass *getApiClass (QString typeIdName)
PyObject *PyTupleGetItem (PyObject *o, int itm)
PyObject *PyListGetItem (PyObject *o, int itm)
int PyTupleCount (PyObject *o)
int PyListCount (PyObject *o)
bool PyIsTuple (PyObject *o)
bool PyIsList (PyObject *o)
bool PyIsUnicode (PyObject *o)
bool PyIsBytes (PyObject *o)
bool PyIsDict (PyObject *o)
bool PyIsBool (PyObject *o)
bool PyIsFloat (PyObject *o)
bool PyIsLong (PyObject *o)
void PyIncRef (PyObject *o)
void PyDecRef (PyObject *o)
void PyTupleSetItem (PyObject *o, int itm, PyObject *val)
PyObject *PyObjectCall (PyObject *fct, PyObject *args)
PyObject *PyTupleNew (int cnt)
PyObject *PyDictNew ()
PyObject *PyDictKeys (PyObject *o)
PyObject *PyDictGetItem (PyObject *dict, PyObject *key)
void PyDictSetItem (PyObject *dict, PyObject *key, PyObject *val)
bool PythonToC_ExecuteGuarded (std::function<void>
    > fct)
void invokePython (std::function<void>
    > fct)
```

```
template<typename R, typename T, typename ...ArgsF, typename ...ArgsT>
R CallNativeFunctionWithTuple (T *pObj, R (T::* f)) ArgsF...
    , std::tuple<ArgsT...> const &t

template<typename R, typename T, typename ...ArgsF, typename ...ArgsT>
R CallNativeHelperFunctionWithTuple (T *pObj, R (*f)) T*, ArgsF...
    , std::tuple<ArgsT...> const &t

template<int pos, class ...Args>
class _PyObjectFunctionToNativeFunction_SetTupleValue
    #include <pythonscriptinterpreter.h>

template<int pos>
class _PyObjectFunctionToNativeFunction_SetTupleValue<pos>
    #include <pythonscriptinterpreter.h>
```

### Public Static Functions

```
void setTupleValue (PyObject*)

template<int pos, class Arg, class ...Args>
class _PyObjectFunctionToNativeFunction_SetTupleValue<pos, Arg, Args...>
    #include <pythonscriptinterpreter.h>
```

### Public Static Functions

```
void setTupleValue (PyObject *tup, Arg arg, Args... args)

template<int pos, class ...Args>
class _PyObjectFunctionToNativeFunction_TupleValueSetter
    #include <pythonscriptinterpreter.h> This class is a workaround for gcc<4.9 bug 55914. It was not able
    to call ...::setTupleValue directly from within the invokePython lambda
```

### Public Functions

```
_PyObjectFunctionToNativeFunction_TupleValueSetter (Args... args)

void set (PyObject *tuple)
```

### Private Members

```
std::function<void (PyObject*)> _set
```

### Private Static Functions

```
void __set (Args... args, PyObject *tuple)

template<typename T, T>
struct CallNativeHelperMethodWithPyObjectArguments
    #include <pythonscriptinterpreter.h>

template<typename T, typename R, typename ... Args, R(*) (T *, Args...) MF> CallNativeH
    #include <pythonscriptinterpreter.h>
```



### Public Static Functions

```
PyObject *call (PyObject *a, PyObject *b)

template<typename T, typename ... Args, void(*) (T *, Args...) MF> CallNativeHelperMeth
#include <pythonscriptinterpreter.h>
```

### Public Static Functions

```
PyObject *call (PyObject *a, PyObject *b)

template<typename T, T>
struct CallNativeMethodWithPyObjectArguments
#include <pythonscriptinterpreter.h>

template<typename T, typename R, typename ... Args, R(T::*)(Args...) const MF> *) (Args
#include <pythonscriptinterpreter.h>
```

### Public Static Functions

```
PyObject *call (PyObject *a, PyObject *b)

template<typename T, typename R, typename ... Args, R(T::*)(Args...) MF> *) (Args...), I
#include <pythonscriptinterpreter.h>
```

### Public Static Functions

```
PyObject *call (PyObject *a, PyObject *b)

template<typename T, typename ... Args, void(T::*)(Args...) const MF> *) (Args...) cons
#include <pythonscriptinterpreter.h>
```

### Public Static Functions

```
PyObject *call (PyObject *a, PyObject *b)

template<typename T, typename ... Args, void(T::*)(Args...) MF> *) (Args...), MF >
#include <pythonscriptinterpreter.h>
```

### Public Static Functions

```
PyObject *call (PyObject *a, PyObject *b)

class Converters
#include <pythonscriptinterpreter.h>
```

### Public Static Functions

```
QString getStringFromPyObject (PyObject *o)
QVariant getVariantFromPyObject (PyObject *o)
int getIntFromPyObject (PyObject *o)
double getDoubleFromPyObject (PyObject *o)
qint64 getInt64FromPyObject (PyObject *o)
quint64 getUInt64FromPyObject (PyObject *o)
bool getBoolFromPyObject (PyObject *o)
QByteArray getByteArrayFromPyObject (PyObject *o)
void *getPointerFromPyObject (PyObject *o)
std::shared_ptr<void> getSharedPointerFromPyObject (PyObject *o)
PyObject *getPyObjectFromString (QString v)
PyObject *getPyObjectFromVariant (QVariant v)
PyObject *getPyObjectFromInt (int v)
PyObject *getPyObjectFromDouble (double v)
PyObject *getPyObjectFromInt64 (qint64 v)
PyObject *getPyObjectFromUInt64 (quint64 v)
PyObject *getPyObjectFromBool (bool v)
PyObject *getPyObjectFromByteArray (QByteArray v)

template<int mode, uint N>
struct FunctionBindTuple
    #include <pythonscriptinterpreter.h>

template<>
struct FunctionBindTuple<0, 0>
    #include <pythonscriptinterpreter.h>
```

### Public Static Functions

```
template<typename R, typename T, typename ...ArgsF, typename ...ArgsT, typename ...Args>
R applyTuple (T *pObj, R (T::*f)) ArgsF...
    , const std::tuple<ArgsT...>&, Args... args

template<uint N>
struct FunctionBindTuple<0, N>
    #include <pythonscriptinterpreter.h>
```

### Public Static Functions

```

template<typename R, typename T, typename ...ArgsF, typename ...ArgsT, typename ...Args>
R applyTuple (T *pObj, R (T::*f)) ArgsF...
    , const std::tuple<ArgsT...> &t, Args... args

template<>
struct FunctionBindTuple<1, 0>
    #include <pythonscriptinterpreter.h>

```

### Public Static Functions

```

template<typename R, typename T, typename ...ArgsF, typename ...ArgsT, typename ...Args>
R applyTuple (T *pObj, R (*f)) T*, ArgsF...
    , const std::tuple<ArgsT...> &t, Args... args

template<uint N>
struct FunctionBindTuple<1, N>
    #include <pythonscriptinterpreter.h>

```

### Public Static Functions

```

template<typename R, typename T, typename ...ArgsF, typename ...ArgsT, typename ...Args>
R applyTuple (T *pObj, R (*f)) T*, ArgsF...
    , const std::tuple<ArgsT...> &t, Args... args

template<class V>
class NativeToPyObject
    #include <pythonscriptinterpreter.h> Helps to return a Python object from a native c++ value (with refer-
    ence ownership)

template<>
class NativeToPyObject<bool>
    #include <pythonscriptinterpreter.h>

```

### Public Static Functions

```

PyObject *toPyObject (bool v)

template<>
class NativeToPyObject<double>
    #include <pythonscriptinterpreter.h>

```

### Public Static Functions

```

PyObject *toPyObject (double v)

template<>
class NativeToPyObject<int>
    #include <pythonscriptinterpreter.h>

```

### Public Static Functions

```
PyObject *toPyObject (int v)

template<>
class NativeToPyObject<QByteArray>
    #include <pythonscriptinterpreter.h>
```

### Public Static Functions

```
PyObject *toPyObject (QByteArray v)

template<>
class NativeToPyObject<qint64>
    #include <pythonscriptinterpreter.h>
```

### Public Static Functions

```
PyObject *toPyObject (qint64 v)

template<typename V>
class NativeToPyObject<QList<V>>
    #include <pythonscriptinterpreter.h>
```

### Public Static Functions

```
PyObject *toPyObject (QList<V> v)

template<typename V>
class NativeToPyObject<QMap<QString, V>>
    #include <pythonscriptinterpreter.h>
```

### Public Static Functions

```
PyObject *toPyObject (QMap<QString, V> v)

template<>
class NativeToPyObject<QString>
    #include <pythonscriptinterpreter.h>
```

### Public Static Functions

```
PyObject *toPyObject (QString v)

template<>
class NativeToPyObject<quint64>
    #include <pythonscriptinterpreter.h>
```

**Public Static Functions**

```

PyObject *toPyObject (quint64 v)
template<>
class NativeToPyObject<QVariant>
    #include <pythonscriptinterpreter.h>

```

**Public Static Functions**

```

PyObject *toPyObject (QVariant v)
template<class V> shared_ptr< const V > >
    #include <pythonscriptinterpreter.h>

```

**Public Static Functions**

```

PyObject *toPyObject (std::shared_ptr<const V> v)
template<class V> shared_ptr< V > >
    #include <pythonscriptinterpreter.h>

```

**Public Static Functions**

```

PyObject *toPyObject (std::shared_ptr<V> v)
template<class V>
class NativeToPyObject<V*>
    #include <pythonscriptinterpreter.h>

```

**Public Static Functions**

```

PyObject *toPyObject (V *v)
template<class R, class ...Args>
class PyObjectFunctionToNativeFunction
    #include <pythonscriptinterpreter.h>

```

**Public Static Functions**

```

std::function<R (Args...)> toNativeFunction
    PyObject *self, PyObject *fct
template<class ...Args>
class PyObjectFunctionToNativeFunction<void, Args...>
    #include <pythonscriptinterpreter.h>

```

### Public Static Functions

```
std::function<void (Args...)> toNativeFunction
    PyObject *self, PyObject *fct

template<class V>
class PyObjectToNative
    #include <pythonscriptinterpreter.h> Helps to return a c++ value from a Python object (borrows the ref-
    erence)

template<>
class PyObjectToNative<bool>
    #include <pythonscriptinterpreter.h>
```

### Public Static Functions

```
bool toNative (PyObject*, PyObject *o)

template<>
class PyObjectToNative<double>
    #include <pythonscriptinterpreter.h>
```

### Public Static Functions

```
double toNative (PyObject*, PyObject *o)

template<>
class PyObjectToNative<int>
    #include <pythonscriptinterpreter.h>
```

### Public Static Functions

```
int toNative (PyObject*, PyObject *o)

template<>
class PyObjectToNative<QByteArray>
    #include <pythonscriptinterpreter.h>
```

### Public Static Functions

```
QByteArray toNative (PyObject*, PyObject *o)

template<>
class PyObjectToNative<qint64>
    #include <pythonscriptinterpreter.h>
```

### Public Static Functions

```

qint64 toNative (PyObject*, PyObject *o)

template<class V>
class PyObjectToNative<QList<V>>
    #include <pythonscriptinterpreter.h>

```

### Public Static Functions

```

QList<V> toNative (PyObject *a, PyObject *o)

template<>
class PyObjectToNative<QString>
    #include <pythonscriptinterpreter.h>

```

### Public Static Functions

```

QString toNative (PyObject*, PyObject *o)

template<>
class PyObjectToNative<quint64>
    #include <pythonscriptinterpreter.h>

```

### Public Static Functions

```

qint64 toNative (PyObject*, PyObject *o)

template<>
class PyObjectToNative<QVariant>
    #include <pythonscriptinterpreter.h>

```

### Public Static Functions

```

QVariant toNative (PyObject*, PyObject *o)

template<class V, class ... Args> function< V(Args...)> > >
    #include <pythonscriptinterpreter.h>

```

### Public Static Functions

```

std::function<V (Args...)> toNative
    PyObject *self, PyObject *io

template<class V> shared_ptr< V > >
    #include <pythonscriptinterpreter.h>

```

### Public Static Functions

```
std::shared_ptr<V> toNative (PyObject*, PyObject *o)

template<class V>
class PyObjectToNative<StringMap<V>>
    #include <pythonscriptinterpreter.h>
```

### Public Static Functions

```
StringMap<V> toNative (PyObject *a, PyObject *o)

template<class V>
class PyObjectToNative<V*>
    #include <pythonscriptinterpreter.h>
```

### Public Static Functions

```
V *toNative (PyObject*, PyObject *o)

template<int pos, typename ...Args>
struct PyObjectTupleToNativeTuple
    #include <pythonscriptinterpreter.h>

template<int pos>
struct PyObjectTupleToNativeTuple<pos>
    #include <pythonscriptinterpreter.h>
```

### Public Static Functions

```
std::tuple toNativeTuple (PyObject*, PyObject*)

template<int pos, class T, class ...Args>
struct PyObjectTupleToNativeTuple<pos, T, Args...>
    #include <pythonscriptinterpreter.h>
```

### Public Static Functions

```
std::tuple<T, Args...> toNativeTuple (PyObject *a, PyObject *b)

template<typename D, typename T, T>
struct ScriptedMethodProxy
    #include <pythonscriptinterpreter.h>

template<typename T, typename R, typename ... Args, std::function< R(Args...)> T::* MF>
    #include <pythonscriptinterpreter.h>
```



## Public Static Functions

```
void setImplementation (T *inst, std::function<R> Args...
    > vfcn)
```

```
PyObject *pyObjectFunctionSetImplementation (PyObject *a, PyObject *b)
```

## 10.2.17 Namespace `sh::search`

**namespace** `sh::search`

Infrastructure for searches in the filesystem.

See `sh::search::SearchManager` for more.

```
class Search : public QObject, public std::enable_shared_from_this<Search>
    #include <search.h> A search object represents one search request the user made.
```

## Public Functions

```
~Search ()
```

```
std::shared_ptr<const sh::filesystem::Eurl> eurl ()
    Returns the eurl containing the search configuration.
```

```
std::shared_ptr<sh::filesystem::FilesystemNode> searchnode ()
    Returns the root node of the search (the one named like ‘Search ...’).
```

```
Search (std::shared_ptr<const filesystem::Eurl> eurl, std::shared_ptr<sh::filesystem::FilesystemNode>
    searchnode)
    See sh::search::SearchManager::requestSearch.
```

```
void search ()
    Start the searching.
```

```
bool isSearching ()
    Returns if the search is currently in progress.
```

## Signals

```
void isSearchingChanged ()
    Triggers when Search::isSearching changes.
```

## Private Functions

```
void search (std::shared_ptr<const filesystem::Eurl> eurl, QString relpath)
```

```
void insertFileItem (std::shared_ptr<const sh::filesystem::Eurl> item, QString relpath,
    sh::filesystem::FilesystemNodeType ftype)
```

```
std::shared_ptr<sh::filesystem::FilesystemNode> getDirItem (std::shared_ptr<const
    sh::filesystem::Eurl> item)
```

### Private Members

```
std::shared_ptr<const sh::filesystem::Eurl> _eurl
std::shared_ptr<sh::filesystem::FilesystemNode> _searchnode
sh::search::SearchConfiguration *_config
QHash<std::shared_ptr<const sh::filesystem::Eurl>, std::shared_ptr<sh::filesystem::FilesystemNode>> _dirs
QMutex _isSearchingMutex
bool _isSearching = false
bool _isSearchAgain = false
```

### class SearchConfiguration

*#include <searchconfiguration.h>* A search configuration.

*Search* configurations contain a list of search criteria and some auxiliary fields. Each search configuration describes what and how a user wants to search (just not where).

### Public Functions

**SearchConfiguration()**

QString **serialize()**

Serialize this search configuration to a string. See also *SearchConfiguration::deserialize*.

### Public Members

QList<std::shared\_ptr<sh::search::SearchCriterion>> **criteria**

bool **showAsTree** = false

### Public Static Functions

*SearchConfiguration* \***deserialize**(QString s)

Deserialize a search configuration from a string. See also *SearchConfiguration::serialize*.

### class SearchCriterion

*#include <searchcriterion.h>* Abstract base class for a search criterion.

Each search criterion implements how a user can filter his files in a search.

Register an implementation, e.g. with ‘REGISTER\_CRITERION\_FACTORY’ of “search/searchcriterionfactoryfromfunction.h”.

Subclassed by *sh::search::criteria::AbstractActionDrivenSearchCriterion*, *sh::search::criteria::AbstractStringSearchCriterion*, *sh::search::criteria::ExtendedAttributeSearchCriterion*, *sh::search::criteria::MtimeSearchCriterion*

## Public Functions

**SearchCriterion** (std::shared\_ptr<sh::search::SearchCriterionFactory> factory)

**~SearchCriterion** ()

QString **serialize** ()

Serializes this criterion to a string. See also *SearchCriterion::deserialize*.

bool **match** (sh::filesystem::Operation \*op, std::shared\_ptr<const sh::filesystem::Eurl> eurl) = 0

Checks if a given file (by eurl) matches this criterion. .

QString **valuedescription** () = 0

Returns a human readable text describing the current criterion shortly and precisely. .

std::shared\_ptr<sh::search::SearchCriterionFactory> **factory** ()

Returns the factory this criterion has created (provides some more metadata).

## Public Static Functions

std::shared\_ptr<sh::search::SearchCriterion> **deserialize** (QString s)

Deserializes this criterion from a string. See also *SearchCriterion::serialize*.

**class SearchCriterionFactory** : public std::enable\_shared\_from\_this<SearchCriterionFactory>  
#include <searchcriterionfactory.h> Abstract base class for a search criterion factory, which constructs some *sh::search::SearchCriterion* instances.

It also provides some control methods and metadata (which is possible since there is a separate factory for each criterion class).

Note: Though possible in some exotic situations, one should not override this class. Override and register *sh::search::SearchCriterion* instead!

Subclassed by *sh::scripting::api::ApiSearchCriterionFactory*, *sh::search::SearchCriterionFactoryFromFunction*

## Public Functions

**SearchCriterionFactory** ()

QString **key** () = 0

Returns the string key for the associated search criterion class. .

QString **description** () = 0

Returns the description string for the associated search criterion class. .

void **editor** (std::shared\_ptr<sh::search::SearchCriterion> c, sh::ui::SearchPanelConfiguration \*panelcfg) = 0

Builds the search config ui in a search panel.

Takes the data from c and populates panelcfg with it.

void **editorupdateconfig** (sh::ui::SearchPanelConfiguration \*panelcfg, std::shared\_ptr<sh::search::SearchCriterion> c) = 0

Updates own data with the content from the search config ui in a search panel.

Takes the data from panelcfg and updates c with it.

std::shared\_ptr<sh::search::SearchCriterion> **construct** () = 0

Constructs an instance of the associated search criterion class. .

```
bool isVisibleFor (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::FilesystemNode>
                    node)
```

Checks if the associated search criterion class should be offered to the user for a given node. .

```
~SearchCriterionFactory ()
```

```
class SearchCriterionFactoryFromFunction : public sh::search::SearchCriterionFactory
#include <searchcriterionfactoryfromfunction.h> A search criterion factory implementation which uses
some additional static methods in the sh::search::SearchCriterion subclasses for control.
```

This is the default search criterion factory. It is typically used indirectly by the 'REGISTER\_CRITERION\_FACTORY' macro from here.

## Public Functions

```
SearchCriterionFactoryFromFunction (QString key, std::function<QString>
> descriptionfctstd::function<voidstd::shared_ptr<sh::search::SearchCriterion>,
sh::ui::SearchPanelConfiguration*> editorfct, std::function<voidsh::ui::SearchPanelConfiguration*,
std::shared_ptr<sh::search::SearchCriterion>> editorupdateconfigfct,
std::function<std::shared_ptr<sh::search::SearchCriterion>std::shared_ptr<sh::search::SearchCriterionFactory>>
constructfct
```

```
QString key ()
```

Returns the string key for the associated search criterion class. .

```
QString description ()
```

Returns the description string for the associated search criterion class. .

```
void editor (std::shared_ptr<sh::search::SearchCriterion> c, sh::ui::SearchPanelConfiguration
*panelcfg)
```

Builds the search config ui in a search panel.

Takes the data from c and populates panelcfg with it.

```
void editorupdateconfig (sh::ui::SearchPanelConfiguration *panelcfg,
std::shared_ptr<sh::search::SearchCriterion> c)
```

Updates own data with the content from the search config ui in a search panel.

Takes the data from panelcfg and updates c with it.

```
std::shared_ptr<SearchCriterion> construct ()
```

Constructs an instance of the associated search criterion class. .

```
bool isVisibleFor (sh::filesystem::Operation *op, std::shared_ptr<sh::filesystem::FilesystemNode>
                    node)
```

Checks if the associated search criterion class should be offered to the user for a given node. .

## Private Members

```
QString _key
```

```
std::function<QString ()> _descriptionfct
```

```
std::function<void (std::shared_ptr<sh::search::SearchCriterion>, sh::ui::SearchPanelConfiguration*)>
    _editorfct
```

```
std::function<void (sh::ui::SearchPanelConfiguration*, std::shared_ptr<sh::search::SearchCriterion>)>
    _editorupdateconfigfct
```

```
std::function< std::shared_ptr< sh::search::SearchCriterion >std::shared_ptr< sh::s
```

```
class SearchFilesystemHandler : public sh::filesystem::FilesystemHandler, public sh::base::Singleton
    #include <searchfilesystemhandler.h> A filesystem handler for presenting filesystem search results.
```

## Public Functions

```
void itemList (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
              sh::filesystem::FilesystemNodeType type, sh::filesystem::FilesystemNodeListEditor
              *list) override

sh::filesystem::FilesystemNodeType getType (sh::filesystem::Operation *op,
                                             std::shared_ptr<const sh::filesystem::Eurl> eurl)
                                             override

qint64 getSize (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl)
               override

QDateTime getMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                   eurl) override

void setMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
              QDateTime time) override

QString getMimeType (sh::filesystem::Operation *op, std::shared_ptr<const
                 sh::filesystem::Eurl> eurl) override

QString getLinkTarget (sh::filesystem::Operation *op, std::shared_ptr<const
                     sh::filesystem::Eurl> eurl) override

bool canGetFileContent (sh::filesystem::Operation *op, std::shared_ptr<const
                      sh::filesystem::Eurl> eurl) override

std::shared_ptr<QIODevice> getFileContent (sh::filesystem::Operation *op,
                                           std::shared_ptr<const sh::filesystem::Eurl>
                                           eurl) override

bool canCreateDirectory (sh::filesystem::Operation *op, std::shared_ptr<const
                        sh::filesystem::Eurl> eurl) override

void createDirectory (sh::filesystem::Operation *op, std::shared_ptr<const
                     sh::filesystem::Eurl> eurl) override

bool canCreateLink (sh::filesystem::Operation *op, std::shared_ptr<const
                   sh::filesystem::Eurl> eurl) override

void createLink (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                eurl, QString target) override

bool canCreateFile (sh::filesystem::Operation *op, std::shared_ptr<const
                  sh::filesystem::Eurl> eurl) override

void createFile (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                eurl, QIODevice *content, HandlerTransfer *handlertransfer) override

bool canDeleteItem (sh::filesystem::Operation *op, std::shared_ptr<const
                  sh::filesystem::Eurl> eurl) override

void deleteItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                eurl) override

bool canRenameItem (sh::filesystem::Operation *op, std::shared_ptr<const
                   sh::filesystem::Eurl> src) override

void renameItem (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> src,
                 QString destpath) override
```

```
QList<std::shared_ptr<sh::actions::AbstractActionItem>> getActions (const
                                                                    QList<std::shared_ptr<const
                                                                    sh::filesystem::Eurl>> eurl) override

void customizeUi (std::shared_ptr<sh::filesystem::FilesystemNode> node, bool *showSearch-
                  Panel) override

void configureItems (sh::filesystem::Operation *op, QList<std::shared_ptr<sh::filesystem::FilesystemNode>>
                    nodes) override
    Configure newly created nodes (e.g. setting another icon or changing the display name).

void itemlist (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
               sh::filesystem::FilesystemNodeType type, sh::filesystem::FilesystemNodeListEditor
               *list) = 0
    Determine a list of subelements in a certain directory.

sh::filesystem::FilesystemNodeType getType (sh::filesystem::Operation *op,
                                             std::shared_ptr<const sh::filesystem::Eurl> eurl) =
                                             0
    Determine if a file is regular, or a dir, or a link, ...

qint64 getSize (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl)
              = 0
    Determine the size of a file.

QDateTime getMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl>
                   eurl) = 0
    Determine the mtime of a file.

void setMtime (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl,
               QDateTime time) = 0
    Set the mtime of a file.

QString getMimetype (sh::filesystem::Operation *op, std::shared_ptr<const
                  sh::filesystem::Eurl> eurl) = 0
    Determine mime type of a file.

QString getLinkTarget (sh::filesystem::Operation *op, std::shared_ptr<const
                      sh::filesystem::Eurl> eurl) = 0
    Resolve a link.

QList<QString> listExtendedAttributes (sh::filesystem::Operation *op,
                                       std::shared_ptr<const sh::filesystem::Eurl>
                                       eurl)
    Fetches the list of available extended attributes for an entry.

qint64 getExtendedAttributeSize (sh::filesystem::Operation *op, std::shared_ptr<const
                                sh::filesystem::Eurl> eurl, QString attribute)
    Returns the size of value for one extended attribute for an entry.

QByteArray getExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
                                sh::filesystem::Eurl> eurl, QString attribute)
    Returns the value for one extended attribute for an entry.

void setExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
                           sh::filesystem::Eurl> eurl, QString attribute, QByteArray
                           value)
    Sets the value for one extended attribute for an entry.

void removeExtendedAttribute (sh::filesystem::Operation *op, std::shared_ptr<const
                              sh::filesystem::Eurl> eurl, QString attribute)
    Removes one extended attributes for an entry.
```

QMap<QString, QString> **getCustomAttributes** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)

Returns the custom attributes for an entry.

In contrast to extended attributes, the custom attributes are not directly stored in the filesystem this way, but handle implementation specific aspects (like permissions).

void **setCustomAttribute** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QString attribute, QString value)

Sets the value for one custom attribute for an entry.

See also getCustomAttributes.

bool **canGetFileContent** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0

Returns if it is allowed to get the content of a certain file.

std::shared\_ptr<QIODevice> **getFileContent** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0

Get the content of a file.

bool **canCreateDirectory** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0

Returns if it is allowed to create subdirectories in a certain directory.

void **createDirectory** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0

Create a directory.

bool **canCreateLink** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0

Returns if it is allowed to create a link in a certain directory.

void **createLink** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QString target) = 0

Create a link.

bool **canCreateFile** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0

Returns if it is allowed to create files in a certain directory.

void **createFile** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QIODevice \*content, HandlerTransfer \*handlertransfer) = 0

Creates a file with some content.

It may use handlertransfer for some better integration, like cancel support and progress monitoring.

bool **canDeleteItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0

Returns if it is allowed to delete a certain file/directory/link/...

void **deleteItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl) = 0

Delete a file/directory/link/...

void **deleteDirectoryIfEmpty** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)

Deletes a directory only if it is empty.

bool **canRenameItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> src) = 0

Returns if it is allowed to rename a certain file, directory, link, ... (see renameItem).



void **renameItem** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> src, QString *destpath*) = 0  
Renames an item to another path.

It can be a simple filename change or also changes in the deeper path segments.

QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> **getActions** (const QList<std::shared\_ptr<const *sh::filesystem::Eurl*>> *eurls*) = 0

Returns a list of actions (see former example) for showing for certain files.

bool **requiresResolvedLinks** ()  
Returns if this handler requires to be used by the engine with resolved links.

void **viewEnteredDirectory** (std::shared\_ptr<const *sh::filesystem::Eurl*>)  
Callback for preparing stuff when the view entered some directory (e.g. for enabling watchers). See also **viewLeftDirectory**.

void **viewLeftDirectory** (std::shared\_ptr<const *sh::filesystem::Eurl*>)  
Callback for preparing stuff when the view left some directory. See also **viewEnteredDirectory**.

bool **isWellKnownScheme** ()  
Returns if the scheme for this handler is a well known one (which external programs typically would understand).

void **search** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*, std::function<void> std::shared\_ptr<const *sh::filesystem::Eurl*>, QList<std::shared\_ptr<*sh::search::SearchCriterion*>>, *sh::filesystem::FilesystemNodeType* ftype > *addfile*, std::function<void> std::shared\_ptr<const *sh::filesystem::Eurl*>> *visitdir*, QList<std::shared\_ptr<*sh::search::SearchCriterion*>> *criteria*)  
Helps searching for files.

Override this method with a behavior identical to the default, if a specialized implementation can be much faster than the default one.

void **customizeUi** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> *node*, bool \**showSearchPanel*)  
Creates a custom gui, which becomes part of the fileview.

*sh::filesystem::FilesystemModel* \***model** ()  
A convenience shortcut to the *sh::filesystem::FilesystemModel* (a singleton).

void **doShutdown** ()  
Executes singleton shutdown.

void **shutdown** ()  
Shutdown down this singleton.

bool **isAlive** ()  
Returns if this singleton is alive (true until its shutdown begins).



## Private Functions

**SearchFilesystemHandler** ()

**class SearchManager** : public QObject, public *sh::base::Singleton*  
*#include <searchmanager.h>* Manager for searches in the filesystem.

## Public Functions

*std::shared\_ptr<sh::search::Search>* **requestSearch** (*std::shared\_ptr<const sh::filesystem::Eurl>* *eurl*,  
 QString *searchconfig*,  
*std::shared\_ptr<sh::filesystem::FilesystemNode>* *searchrootnode*)

Requests a search.

void **removeSearch** (*std::shared\_ptr<sh::search::Search>* *search*)  
 Removes a search.

*std::shared\_ptr<sh::search::Search>* **findSearch** (*std::shared\_ptr<const sh::filesystem::Eurl>* *eurl*)

Returns the *Search* instance for an eurl for a search root node.

void **addFactory** (int *index*, *std::shared\_ptr<SearchCriterionFactory>* *f*)  
 Adds a search criterion factory.

QList<*std::shared\_ptr<SearchCriterionFactory>*> **factories** ()  
 Returns the sorted list of search criterion factories.

**~SearchManager** ()

void **doShutdown** ()  
 Executes singleton shutdown.

void **shutdown** ()  
 Shutdown down this singleton.

bool **isAlive** ()  
 Returns if this singleton is alive (true until its shutdown begins).

## Public Static Attributes

**const** int **REGISTER\_SEARCHCRITERION\_INDEX\_CORE** = 1000000  
 Base value for display indexes of core (i.e. very important) search criteria.  
 Used in *SearchManager::addFactory*.

**const** int **REGISTER\_SEARCHCRITERION\_INDEX\_NORMAL** = 2000000  
 Base value for display indexes of search criteria with normal importance.  
 Used in *SearchManager::addFactory*.

**const** int **REGISTER\_SEARCHCRITERION\_INDEX\_EXOTIC** = 3000000  
 Base value for display indexes of exotic search criteria.  
 Used in *SearchManager::addFactory*.

## Private Functions

**SearchManager** ( )

## Private Members

QList<std::shared\_ptr<*sh::search::Search*>> **\_searches**

QList<std::weak\_ptr<*sh::search::Search*>> **\_weak\_searches**

QMap<int, std::shared\_ptr<*SearchCriterionFactory*>> **\_factories**

## namespace criteria

Implementations of search criteria.

Subclasses of *sh::search::SearchCriterion*.

**class AbstractActionDrivenSearchCriterion** : public *sh::search::SearchCriterion*  
#include <abstractactiondrivensearchcriterion.h> Abstract search criterion filtering by something which is configured in a wizard-like way by an action execution.

This provides an easy yet powerful programming interface, e.g. for usage in scripting.

Subclassed by *sh::scripting::api::ApiSearchCriterion*

## Public Functions

**AbstractActionDrivenSearchCriterion** (std::shared\_ptr<*sh::search::SearchCriterionFactory*>  
factory)

bool **match** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)  
Checks if a given file (by eurl) matches this criterion. .

bool **match2** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)  
= 0

QString **valuedescription** ()  
Returns a human readable text describing the current criterion shortly and precisely. .

QString **serialize** ()  
Serializes this criterion to a string. See also *SearchCriterion::deserialize*.

std::shared\_ptr<*sh::search::SearchCriterionFactory*> **factory** ()  
Returns the factory this criterion has created (provides some more metadata).

## Public Members

QStringList **searchspec**

## Public Static Functions

```
void _createEditor (std::shared_ptr<sh::search::SearchCriterion>                cfg,
                   sh::ui::SearchPanelConfiguration *panelcfg)
void _editorUpdateConfiguration (sh::ui::SearchPanelConfiguration *panelcfg,
                                  std::shared_ptr<sh::search::SearchCriterion> c)
std::shared_ptr<sh::search::SearchCriterion> deserialize (QString s)
    Deserializes this criterion from a string. See also SearchCriterion::serialize.
```

## Friends

```
friend class ActionAbstractActionDrivenSearchCriterionConfigure
class AbstractStringSearchCriterion : public sh::search::SearchCriterion
    #include <abstractstringsearchcriterion.h> Abstract search criterion filtering by some string value
    (providing different modes like exact, fuzzy, regexp, ...).
    Subclassed by sh::search::criteria::FileContentSearchCriterion, sh::search::criteria::FilenameSearchCriterion
```

## Public Types

```
enum Mode
    Values:
        enumerator Simple
        enumerator Exact
        enumerator RegExp
        enumerator Fuzzy
```

## Public Functions

```
AbstractStringSearchCriterion (std::shared_ptr<sh::search::SearchCriterionFactory>
                                factory)
bool match (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl)
    Checks if a given file (by eurl) matches this criterion. .
QString valuedescription ()
    Returns a human readable text describing the current criterion shortly and precisely. .
QString serialize ()
    Serializes this criterion to a string. See also SearchCriterion::deserialize.
std::shared_ptr<sh::search::SearchCriterionFactory> factory ()
    Returns the factory this criterion has created (provides some more metadata).
```

## Public Members

QString **string**

*Mode* **mode** = *Mode::Simple*

## Public Static Functions

void **\_createEditor** (std::shared\_ptr<sh::search::SearchCriterion> *cfg*,  
ui::SearchPanelConfiguration \*panelcfg)

void **\_editorUpdateConfiguration** (sh::ui::SearchPanelConfiguration \*panelcfg,  
std::shared\_ptr<sh::search::SearchCriterion> c)

int **levenshteinSubstringDistance** (QString *needle*, QString *haystack*)

double **relativeLevenshteinPartsSubstringDistance** (QString *needle*, QString  
*haystack*)

std::shared\_ptr<sh::search::SearchCriterion> **deserialize** (QString *s*)  
Deserializes this criterion from a string. See also *SearchCriterion::serialize*.

## Public Static Attributes

const QStringList **ModeCodes**

**class ActionAbstractActionDrivenSearchCriterionConfigure** : public sh::actions::ActionActionItem  
#include <abstractactiondrivensearchcriterion.h>

## Public Functions

**ActionAbstractActionDrivenSearchCriterionConfigure** (std::shared\_ptr<AbstractActionDrivenSearchCriterion> *cfg*, QMutex \**mutex*)

void **action** (sh::actions::ActionExecutionInfo \**info*)  
The action implementation, i.e. what the actions should actually do.

void **execute** ()  
Executes this action.

void **execute** (sh::actions::ActionExecutionInfo \**info*)  
Executes this action.

QKeySequence **shortcuthint** ()  
Returns the keyboard shortcut for triggering this action.

bool **shortcuthintTriggersOnCurrentDirectory** ()  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

void **setShortcuthint** (QKeySequence *shortcut*, bool *triggersOnCurrentDirectory* = false)  
Sets the keyboard shortcut for triggering this action.

QString **text** ()  
Returns the displayed text for this action.

QString **icon** ()  
Returns the icon for this action.

bool **enabled** ()  
Checks if this action is enabled.

bool **isChecked** ()  
Checks if this action is checked (has a cross in the ui).

bool **isCheckable** ()  
Checks if this action is checkable (can have a cross in the ui).

int **defaultActionPrecedence** () **const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

void **setText** (QString *text*)  
Sets the displayed text.

void **setIcon** (QString *icon*)  
Sets the icon.

void **setEnabled** (bool *enabled*)  
Sets if the item is enabled.

void **setChecked** (bool *checked*)  
Sets if the item is checked (has a cross in the ui).

void **setVisible** (bool *visible*)  
Sets the visibility of this item.

bool **visible** ()  
Checks the visibility of this item (non-recursively).

std::weak\_ptr<AbstractActionItem> **parentAction** ()  
Returns the parent action, if it is added to a container.

void **\_setParentAction** (std::shared\_ptr<AbstractActionItem> *parent*)  
Sets the parent action. .

void **initializeAsync** (std::function<void>  
> *oninitialized* = 0) Asynchronously initializes the action.

void **initializeSync** ()  
Synchronously initializes the action.

bool **isInitialized** ()  
Checks if this action is initialized.

bool **isInitializing** ()  
Checks if this action is initializing.

## Signals

void **changed** ()  
Emits when some data changed.

## Private Members

std::shared\_ptr<*AbstractActionDrivenSearchCriterion*> **\_cfg**

QMutex **\*\_mutex**

**class ExtendedAttributeSearchCriterion : public *sh::search::SearchCriterion***  
*#include <extendedattributearchcriterion.h>* *Search* criterion filtering by extended attributes.

## Public Types

**enum Mode**

*Values:*

**enumerator None**

**enumerator Simple**

**enumerator RegExp**

**enumerator Fuzzy**

## Public Functions

**ExtendedAttributeSearchCriterion** (std::shared\_ptr<*sh::search::SearchCriterionFactory*>  
*factory*)

bool **match** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)  
Checks if a given file (by eurl) matches this criterion. .

QString **valuedescription** ()  
Returns a human readable text describing the current criterion shortly and precisely. .

QString **serialize** ()  
Serializes this criterion to a string. See also *SearchCriterion::deserialize*.

std::shared\_ptr<*sh::search::SearchCriterionFactory*> **factory** ()  
Returns the factory this criterion has created (provides some more metadata).

## Public Members

*Mode* **keymode** = *Mode::None*

*Mode* **valuemode** = *Mode::Simple*

QString **key**

QString **value**

## Public Static Functions

```
void createEditor (std::shared_ptr<sh::search::SearchCriterion>                cfg,
                  sh::ui::SearchPanelConfiguration *panelcfg)
void editorUpdateConfiguration (sh::ui::SearchPanelConfiguration *panelcfg,
                                std::shared_ptr<sh::search::SearchCriterion> c)
void doInitialize ()
void doShutdown ()
std::shared_ptr<sh::search::SearchCriterion> deserialize (QString s)
    Deserializes this criterion from a string. See also SearchCriterion::serialize.
```

## Public Static Attributes

```
const QStringList ModeCodes
```

```
class FileContentSearchCriterion : public sh::search::criteria::AbstractStringSearchCriterion
    #include <filecontentsearchcriterion.h> Search criterion filtering by file contents.
```

## Public Types

```
enum Mode
    Values:
        enumerator Simple
        enumerator Exact
        enumerator RegExp
        enumerator Fuzzy
```

## Public Functions

```
FileContentSearchCriterion (std::shared_ptr<sh::search::SearchCriterionFactory>
                             factory)
bool match (sh::filesystem::Operation *op, std::shared_ptr<const sh::filesystem::Eurl> eurl)
    Checks if a given file (by eurl) matches this criterion. .
QString valuedescription ()
    Returns a human readable text describing the current criterion shortly and precisely. .
QString serialize ()
    Serializes this criterion to a string. See also SearchCriterion::deserialize.
std::shared_ptr<sh::search::SearchCriterionFactory> factory ()
    Returns the factory this criterion has created (provides some more metadata).
```

## Public Members

QString **string**

*Mode* **mode** = *Mode::Simple*

## Public Static Functions

void **createEditor** (std::shared\_ptr<*sh::search::SearchCriterion*> *cfg*,  
                    *sh::ui::SearchPanelConfiguration* \*panelcfg)

void **editorUpdateConfiguration** (*sh::ui::SearchPanelConfiguration* \*panelcfg,  
                                  std::shared\_ptr<*sh::search::SearchCriterion*> c)

void **doInitialize** ()

void **doShutdown** ()

void **\_createEditor** (std::shared\_ptr<*sh::search::SearchCriterion*> *cfg*,  
                    *ui::SearchPanelConfiguration* \*panelcfg)

void **\_editorUpdateConfiguration** (*sh::ui::SearchPanelConfiguration* \*panelcfg,  
                                  std::shared\_ptr<*sh::search::SearchCriterion*> c)

int **levenshteinSubstringDistance** (QString *needle*, QString *haystack*)

double **relativeLevenshteinPartsSubstringDistance** (QString *needle*, QString  
  *haystack*)

std::shared\_ptr<*sh::search::SearchCriterion*> **deserialize** (QString s)  
    Deserializes this criterion from a string. See also *SearchCriterion::serialize*.

## Public Static Attributes

const QStringList **ModeCodes**

**class FilenameSearchCriterion**: public *sh::search::criteria::AbstractStringSearchCriterion*  
    #include <filenamesearchcriterion.h> *Search* criterion filtering by file names.

## Public Types

enum **Mode**

*Values:*

enumerator **Simple**

enumerator **Exact**

enumerator **RegExp**

enumerator **Fuzzy**



## Public Functions

**FilenameSearchCriterion** (std::shared\_ptr<sh::search::SearchCriterionFactory> *factory*)

bool **match** (sh::filesystem::Operation \*op, std::shared\_ptr<const sh::filesystem::Eurl> eurl)  
Checks if a given file (by eurl) matches this criterion. .

QString **valuedescription** ()  
Returns a human readable text describing the current criterion shortly and precisely. .

QString **serialize** ()  
Serializes this criterion to a string. See also *SearchCriterion::deserialize*.

std::shared\_ptr<sh::search::SearchCriterionFactory> **factory** ()  
Returns the factory this criterion has created (provides some more metadata).

## Public Members

QString **string**  
*Mode mode = Mode::Simple*

## Public Static Functions

void **createEditor** (std::shared\_ptr<sh::search::SearchCriterion> *cfg*,  
sh::ui::SearchPanelConfiguration \*panelcfg)

void **editorUpdateConfiguration** (sh::ui::SearchPanelConfiguration \*panelcfg,  
std::shared\_ptr<sh::search::SearchCriterion> c)

void **doInitialize** ()

void **doShutdown** ()

void **\_createEditor** (std::shared\_ptr<sh::search::SearchCriterion> *cfg*,  
sh::ui::SearchPanelConfiguration \*panelcfg)

void **\_editorUpdateConfiguration** (sh::ui::SearchPanelConfiguration \*panelcfg,  
std::shared\_ptr<sh::search::SearchCriterion> c)

int **levenshteinSubstringDistance** (QString *needle*, QString *haystack*)

double **relativeLevenshteinPartsSubstringDistance** (QString *needle*, QString *haystack*)

std::shared\_ptr<sh::search::SearchCriterion> **deserialize** (QString s)  
Deserializes this criterion from a string. See also *SearchCriterion::serialize*.

## Public Static Attributes

const QStringList **ModeCodes**

**class MtimeSearchCriterion**: public sh::search::SearchCriterion  
*#include <mtimesearchcriterion.h>* *Search* criterion filtering by file modification times.

## Public Functions

**MtimeSearchCriterion** (std::shared\_ptr<sh::search::SearchCriterionFactory> *factory*)

bool **match** (sh::filesystem::Operation \**op*, std::shared\_ptr<const sh::filesystem::Eurl> *eurl*)  
Checks if a given file (by eurl) matches this criterion. .

QString **valuedescription** ()  
Returns a human readable text describing the current criterion shortly and precisely. .

QString **serialize** ()  
Serializes this criterion to a string. See also *SearchCriterion::deserialize*.

std::shared\_ptr<sh::search::SearchCriterionFactory> **factory** ()  
Returns the factory this criterion has created (provides some more metadata).

## Public Members

QDateTime **from**  
QDateTime **to**

## Public Static Functions

void **createEditor** (std::shared\_ptr<sh::search::SearchCriterion> *cfg*,  
sh::ui::SearchPanelConfiguration \**panelcfg*)

void **editorUpdateConfiguration** (sh::ui::SearchPanelConfiguration \**panelcfg*,  
std::shared\_ptr<sh::search::SearchCriterion> *c*)

void **doInitialize** ()

void **doShutdown** ()

std::shared\_ptr<sh::search::SearchCriterion> **deserialize** (QString *s*)  
Deserializes this criterion from a string. See also *SearchCriterion::serialize*.

## 10.2.18 Namespace sh::search::criteria

namespace *sh::search::criteria*

Implementations of search criteria.

Subclasses of *sh::search::SearchCriterion*.

**class AbstractActionDrivenSearchCriterion** : public *sh::search::SearchCriterion*  
*#include <abstractactiondrivensearchcriterion.h>* Abstract search criterion filtering by something which is configured in a wizard-like way by an action execution.

This provides an easy yet powerful programming interface, e.g. for usage in scripting.

Subclassed by *sh::scripting::api::ApiSearchCriterion*

## Public Functions

**AbstractActionDrivenSearchCriterion** (std::shared\_ptr<sh::search::SearchCriterionFactory> factory)

bool **match** (sh::filesystem::Operation \*op, std::shared\_ptr<const sh::filesystem::Eurl> eurl)  
Checks if a given file (by eurl) matches this criterion. .

bool **match2** (sh::filesystem::Operation \*op, std::shared\_ptr<const sh::filesystem::Eurl> eurl) = 0

QString **valuedescription** ()  
Returns a human readable text describing the current criterion shortly and precisely. .

QString **serialize** ()  
Serializes this criterion to a string. See also *SearchCriterion::deserialize*.

std::shared\_ptr<sh::search::SearchCriterionFactory> **factory** ()  
Returns the factory this criterion has created (provides some more metadata).

## Public Members

QStringList **searchspec**

## Public Static Functions

void **\_createEditor** (std::shared\_ptr<sh::search::SearchCriterion> cfg,  
sh::ui::SearchPanelConfiguration \*panelcfg)

void **\_editorUpdateConfiguration** (sh::ui::SearchPanelConfiguration \*panelcfg,  
std::shared\_ptr<sh::search::SearchCriterion> c)

std::shared\_ptr<sh::search::SearchCriterion> **deserialize** (QString s)  
Deserializes this criterion from a string. See also *SearchCriterion::serialize*.

## Friends

**friend class** ActionAbstractActionDrivenSearchCriterionConfigure

**class AbstractStringSearchCriterion** : public sh::search::SearchCriterion  
#include <abstractstringsearchcriterion.h> Abstract search criterion filtering by some string value (providing different modes like exact, fuzzy, regexp, ...).

Subclassed by sh::search::criteria::FileContentSearchCriterion, sh::search::criteria::FilenameSearchCriterion

## Public Types

**enum Mode**

Values:

enumerator Simple

enumerator Exact

enumerator RegExp

enumerator Fuzzy

## Public Functions

**AbstractStringSearchCriterion** (std::shared\_ptr<sh::search::SearchCriterionFactory> factory)

bool **match** (sh::filesystem::Operation \*op, std::shared\_ptr<const sh::filesystem::Eurl> eurl)  
Checks if a given file (by eurl) matches this criterion. .

QString **valuedescription** ()  
Returns a human readable text describing the current criterion shortly and precisely. .

QString **serialize** ()  
Serializes this criterion to a string. See also *SearchCriterion::deserialize*.

std::shared\_ptr<sh::search::SearchCriterionFactory> **factory** ()  
Returns the factory this criterion has created (provides some more metadata).

## Public Members

QString **string**  
*Mode mode = Mode::Simple*

## Public Static Functions

void **\_createEditor** (std::shared\_ptr<sh::search::SearchCriterion> c, sh::ui::SearchPanelConfiguration \*panelcfg)

void **\_editorUpdateConfiguration** (sh::ui::SearchPanelConfiguration \*panelcfg, std::shared\_ptr<sh::search::SearchCriterion> c)

int **levenshteinSubstringDistance** (QString needle, QString haystack)

double **relativeLevenshteinPartsSubstringDistance** (QString needle, QString haystack)

std::shared\_ptr<sh::search::SearchCriterion> **deserialize** (QString s)  
Deserializes this criterion from a string. See also *SearchCriterion::serialize*.

## Public Static Attributes

const QStringList **ModeCodes**

**class ActionAbstractActionDrivenSearchCriterionConfigure** : public sh::actions::ActionActionItem  
*#include <abstractactiondrivensearchcriterion.h>*

## Public Functions

**ActionAbstractActionDrivenSearchCriterionConfigure** (std::shared\_ptr<AbstractActionDrivenSearchCriterion> c, QMutex \*mutex)

void **action** (sh::actions::ActionExecutionInfo \*info)  
The action implementation, i.e. what the actions should actually do.

void **execute** ()  
Executes this action.

void **execute** (sh::actions::ActionExecutionInfo \*info)  
Executes this action.

**QKeySequence shortcutHint ()**  
Returns the keyboard shortcut for triggering this action.

**bool shortcutHintTriggersOnCurrentDirectory ()**  
Checks if using the keyboard shortcut shall trigger the action on the current directory (or on the entry selection).

**void setShortcutHint (QKeySequence shortcut, bool triggersOnCurrentDirectory = false)**  
Sets the keyboard shortcut for triggering this action.

**QString text ()**  
Returns the displayed text for this action.

**QString icon ()**  
Returns the icon for this action.

**bool enabled ()**  
Checks if this action is enabled.

**bool isChecked ()**  
Checks if this action is checked (has a cross in the ui).

**bool isCheckable ()**  
Checks if this action is checkable (can have a cross in the ui).

**int defaultActionPrecedence () const**  
Returns the precedence for becoming the default action of a parent submenu.  
  
This e.g. leads to a bold label.  
  
The action with the highest value becomes the default. Only values >0 will be considered. See [ActionDefaultPrecedenceValues](#) for reference values.

**void setText (QString text)**  
Sets the displayed text.

**void setIcon (QString icon)**  
Sets the icon.

**void setEnabled (bool enabled)**  
Sets if the item is enabled.

**void setChecked (bool checked)**  
Sets if the item is checked (has a cross in the ui).

**void setVisible (bool visible)**  
Sets the visibility of this item.

**bool visible ()**  
Checks the visibility of this item (non-recursively).

**std::weak\_ptr<AbstractActionItem> parentAction ()**  
Returns the parent action, if it is added to a container.

**void \_setParentAction (std::shared\_ptr<AbstractActionItem> parent)**  
Sets the parent action. .

**void initializeAsync (std::function<void>  
> oninitialized = 0)** Asynchronously initializes the action.

**void initializeSync ()**  
Synchronously initializes the action.

bool **isInitialized**()  
Checks if this action is initialized.

bool **isInitializing**()  
Checks if this action is initializing.

## Signals

void **changed**()  
Emits when some data changed.

## Private Members

std::shared\_ptr<*AbstractActionDrivenSearchCriterion*> **\_cfg**

QMutex \*\_**mutex**

**class ExtendedAttributeSearchCriterion** : public *sh::search::SearchCriterion*  
*#include <extendedattributesearchcriterion.h>* *Search* criterion filtering by extended attributes.

## Public Types

enum **Mode**

*Values:*

enumerator **None**

enumerator **Simple**

enumerator **RegExp**

enumerator **Fuzzy**

## Public Functions

**ExtendedAttributeSearchCriterion**(std::shared\_ptr<*sh::search::SearchCriterionFactory*>  
*factory*)

bool **match**(*sh::filesystem::Operation* \**op*, std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*)  
Checks if a given file (by eurl) matches this criterion. .

QString **valuedescription**()  
Returns a human readable text describing the current criterion shortly and precisely. .

QString **serialize**()  
Serializes this criterion to a string. See also *SearchCriterion::deserialize*.

std::shared\_ptr<*sh::search::SearchCriterionFactory*> **factory**()  
Returns the factory this criterion has created (provides some more metadata).

## Public Members

*Mode* **keymode** = *Mode::None*

*Mode* **valuemode** = *Mode::Simple*

QString **key**

QString **value**

## Public Static Functions

void **createEditor** (std::shared\_ptr<*sh::search::SearchCriterion*> *cfg*,  
*sh::ui::SearchPanelConfiguration* \*panelcfg)

void **editorUpdateConfiguration** (*sh::ui::SearchPanelConfiguration* \*panelcfg,  
std::shared\_ptr<*sh::search::SearchCriterion*> c)

void **doInitialize** ()

void **doShutdown** ()

std::shared\_ptr<*sh::search::SearchCriterion*> **deserialize** (QString s)  
Deserializes this criterion from a string. See also *SearchCriterion::serialize*.

## Public Static Attributes

const QStringList **ModeCodes**

**class FileContentSearchCriterion** : public *sh::search::criteria::AbstractStringSearchCriterion*  
*#include <filecontentsearchcriterion.h>* *Search* criterion filtering by file contents.

## Public Types

enum **Mode**

*Values:*

enumerator **Simple**

enumerator **Exact**

enumerator **RegExp**

enumerator **Fuzzy**

## Public Functions

**FileContentSearchCriterion** (std::shared\_ptr<*sh::search::SearchCriterionFactory*> *factory*)

bool **match** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)  
Checks if a given file (by eurl) matches this criterion. .

QString **valuedescription** ()  
Returns a human readable text describing the current criterion shortly and precisely. .

QString **serialize** ()  
Serializes this criterion to a string. See also *SearchCriterion::deserialize*.

`std::shared_ptr<sh::search::SearchCriterionFactory> factory ()`  
Returns the factory this criterion has created (provides some more metadata).

## Public Members

`QString string`  
*Mode mode = Mode::Simple*

## Public Static Functions

`void createEditor (std::shared_ptr<sh::search::SearchCriterion> cfg,  
sh::ui::SearchPanelConfiguration *panelcfg)`  
`void editorUpdateConfiguration (sh::ui::SearchPanelConfiguration *panelcfg,  
std::shared_ptr<sh::search::SearchCriterion> c)`  
`void doInitialize ()`  
`void doShutdown ()`  
`void _createEditor (std::shared_ptr<sh::search::SearchCriterion> cfg,  
ui::SearchPanelConfiguration *panelcfg)`  
`void _editorUpdateConfiguration (sh::ui::SearchPanelConfiguration *panelcfg,  
std::shared_ptr<sh::search::SearchCriterion> c)`  
`int levenshteinSubstringDistance (QString needle, QString haystack)`  
`double relativeLevenshteinPartsSubstringDistance (QString needle, QString haystack)`  
`std::shared_ptr<sh::search::SearchCriterion> deserialize (QString s)`  
Deserializes this criterion from a string. See also *SearchCriterion::serialize*.

## Public Static Attributes

`const QStringList ModeCodes`

`class FilenameSearchCriterion: public sh::search::criteria::AbstractStringSearchCriterion`  
*#include <filenamesearchcriterion.h> Search criterion filtering by file names.*

## Public Types

`enum Mode`  
*Values:*  
`enumerator Simple`  
`enumerator Exact`  
`enumerator RegExp`  
`enumerator Fuzzy`



## Public Functions

**FilenameSearchCriterion** (std::shared\_ptr<sh::search::SearchCriterionFactory> *factory*)

bool **match** (sh::filesystem::Operation \**op*, std::shared\_ptr<const sh::filesystem::Eurl> *eurl*)  
Checks if a given file (by eurl) matches this criterion. .

QString **valuedescription** ()

Returns a human readable text describing the current criterion shortly and precisely. .

QString **serialize** ()

Serializes this criterion to a string. See also *SearchCriterion::deserialize*.

std::shared\_ptr<sh::search::SearchCriterionFactory> **factory** ()

Returns the factory this criterion has created (provides some more metadata).

## Public Members

QString **string**

*Mode mode = Mode::Simple*

## Public Static Functions

void **createEditor** (std::shared\_ptr<sh::search::SearchCriterion> *sc*,  
sh::ui::SearchPanelConfiguration \**panelcfg*)

void **editorUpdateConfiguration** (sh::ui::SearchPanelConfiguration \**panelcfg*,  
std::shared\_ptr<sh::search::SearchCriterion> *c*)

void **doInitialize** ()

void **doShutdown** ()

void **\_createEditor** (std::shared\_ptr<sh::search::SearchCriterion> *sc*,  
ui::SearchPanelConfiguration \**panelcfg*)

void **\_editorUpdateConfiguration** (sh::ui::SearchPanelConfiguration \**panelcfg*,  
std::shared\_ptr<sh::search::SearchCriterion> *c*)

int **levenshteinSubstringDistance** (QString *needle*, QString *haystack*)

double **relativeLevenshteinPartsSubstringDistance** (QString *needle*, QString  
*haystack*)

std::shared\_ptr<sh::search::SearchCriterion> **deserialize** (QString *s*)  
Deserializes this criterion from a string. See also *SearchCriterion::serialize*.

## Public Static Attributes

const QStringList **ModeCodes**

**class MtimeSearchCriterion: public sh::search::SearchCriterion**  
*#include <mtimesearchcriterion.h>* *Search* criterion filtering by file modification times.

## Public Functions

**MtimeSearchCriterion** (std::shared\_ptr<sh::search::SearchCriterionFactory> *factory*)

bool **match** (sh::filesystem::Operation \**op*, std::shared\_ptr<const sh::filesystem::Eurl> *eurl*)  
Checks if a given file (by eurl) matches this criterion. .

QString **valuedescription** ()  
Returns a human readable text describing the current criterion shortly and precisely. .

QString **serialize** ()  
Serializes this criterion to a string. See also *SearchCriterion::deserialize*.

std::shared\_ptr<sh::search::SearchCriterionFactory> **factory** ()  
Returns the factory this criterion has created (provides some more metadata).

## Public Members

QDateTime **from**  
QDateTime **to**

## Public Static Functions

void **createEditor** (std::shared\_ptr<sh::search::SearchCriterion> *cfg*,  
sh::ui::SearchPanelConfiguration \**panelcfg*)

void **editorUpdateConfiguration** (sh::ui::SearchPanelConfiguration \**panelcfg*,  
std::shared\_ptr<sh::search::SearchCriterion> *c*)

void **doInitialize** ()

void **doShutdown** ()

std::shared\_ptr<sh::search::SearchCriterion> **deserialize** (QString *s*)  
Deserializes this criterion from a string. See also *SearchCriterion::serialize*.

## 10.2.19 Namespace sh::settings

### namespace sh::settings

Infrastructure for settings, so everything you see in ‘Manage saved settings’.

See the abstract base class *sh::settings::Setting* and *sh::settings::SettingsManager* for more.

## Enums

### enum SettingGroup

Groups of settings.

A choice here does not make a logical impact. It is merely a matter of graphical presentation.

*Values:*

enumerator GLOBAL = 1  
enumerator GUI = 2  
enumerator FILEHANDLING = 3

```

enumerator DATAVIEW = 4
enumerator BEHAVIOR = 5
enumerator SPECIAL = 6

```

## **class ProfileNode**

*#include <profilenode.h>* One entry in a Shallot settings profile, so one item as selectable in the ‘Manage settings’ dialog.

### **Public Functions**

#### **ProfileNode ()**

Constructed only by the infrastructure and made available otherwise.

#### **QString nodeId ()**

An identifier name.

#### **void setNodeId (QString nodeId)**

Sets the identifier name.

#### **std::shared\_ptr<const filesystem::Eurl> eurl ()**

The eurl of the directory, or 0 for global profile nodes.

#### **void setEurl (std::shared\_ptr<const filesystem::Eurl> eurl)**

Sets the eurl.

#### **QString profilename ()**

The profile name.

#### **void setProfilename (QString profilename)**

Sets the profile name.

#### **QStringList inheritsFrom ()**

List of profile names from which this node inherits.

#### **void setInheritsFrom (QStringList profilenames)**

Sets list of profile names from which this node inherits.

#### **bool withSubfolders ()**

If this node also applies recursively on subfolders.

#### **void setWithSubfolders (bool v)**

Sets if this node also applies recursively on subfolders.

#### **void setSetting (QString name, QString value, int onlyInFileview = -1)**

Stores a value for a setting for applying it later.

#### **QList<QString> getSettingNames ()**

List of settings names which are set in this node.

#### **QString getSettingValue (QString name)**

Gets the stored value of a setting by setting name.

#### **int getSettingOnlyInFileviewIndex (QString name)**

Gets if a setting applies to a certain fileview or to the entire window.

### Private Members

QMap<QString, QString> **\_values**  
QMap<QString, int> **\_valuesOnlyInFileviewIndex**  
std::shared\_ptr<const *sh::filesystem::Eurl*> **\_eurl**  
QString **\_profilename**  
QStringList **\_inheritsFrom**  
bool **\_withSubfolders**  
QString **\_nodeId**

### class Setting

*#include <setting.h>* Abstract base class for a Shallot setting.

Those have the greatest flexibility. They have to be stored manually by the user with many options. See Shallot documentation for details.

Use *sh::settings::SettingsRegistration* for registering an implementation.

Subclassed by *sh::scripting::api::ApiSetting*, *sh::settings::common::FileDetailsPanelVisible*,  
*sh::settings::common::FileViewIconListThumbDimension*, *sh::settings::common::FileViewPath*,  
*sh::settings::common::FileViewViewmode*, *sh::settings::common::JumpbarMode*,  
*sh::settings::common::NumberOfFilePanels*, *sh::settings::common::ShowHiddenFiles*,  
*sh::settings::common::ShowTree*, *sh::settings::common::SizeFormattingMode*,  
*sh::settings::common::StickyTreeview*, *sh::settings::common::WindowTitle*

### Public Functions

#### Setting()

Is (for subclasses) intended to be directly constructed from everywhere or by registering a factory somewhere.

#### ~Setting()

#### QString name() = 0

Gets the internal name.

#### QString description() = 0

Gets the description text.

#### SettingGroup group() = 0

Gets the group;.

#### bool isAdvancedSetting() = 0

Is this an advanced setting?

#### void setValue(*sh::ui::FileView\**, QString)

Called from Shallot core when the value was set (for a per-fileview setting).

#### void setValue(QString)

Called from Shallot core when the value was set (for a not-per-fileview setting).

#### bool isGlobal() = 0

Does this setting apply globally or just for a certain subtree of nodes?

#### bool isPerFileview() = 0

Does this setting apply for each fileview individually or for the complete main window?

QString **getValue** (*sh::ui::FileView \*filelist*) = 0

Get the currently set value.

QString **valueDescription** (QString *value*)

Gets a human readable description text for a value.

## Public Static Functions

QString **getGroupDescription** (int *g*)

Low-level function which gets the description text of a group.

**class SettingsManager** : public QObject, public *sh::base::Singleton*  
*#include <settingsmanager.h>* Manager for Shallot settings.

This is the more flexible way of shallot customization. It includes everything you see in the ‘Manage saved settings’ or ‘Save settings’.

See *sh::settings::Setting* for more.

## Public Functions

void **applyPerEurlSettingsToFileView** (*sh::ui::FileView \*list*)

Applies all stored settings, which are per-eurl to a fileview.

void **applyGlobalSettingsToFileView** (*sh::ui::FileView \*list*)

Applies all stored settings, which are global to a fileview.

void **applyPerEurlSettingsToMainWindow** ()

Applies all stored settings, which are per-eurl to main window.

void **applyGlobalSettingsToMainWindow** ()

Applies all stored settings, which are global to a main window.

QList<*sh::settings::ProfileNode\**> **getNodesForProfile** (QString *profile*)

Returns a list of all stored settings (as *ProfileNodes*) for a given profile.

*sh::settings::ProfileNode\** **getNodeById** (QString *nodeId*)

Searches and returns a *ProfileNode* by id.

std::shared\_ptr<*sh::settings::Setting*> **getSettingByName** (QString *name*)

Searches and returns a *Setting* by name.

QList<std::shared\_ptr<*sh::settings::Setting*>> **getAllSettings** ()

Returns a list of all available settings, primarily sorted by group.

QStringList **getProfileList** ()

Returns a list of all existing profiles.

void **removeProfile** (QString *profile*)

Removes a profile (including all *ProfileNodes* inside it).

void **removeNode** (QString *nodeId*)

Removes a *ProfileNode* by id.

void **storeProfile** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*, QString *filename*,  
 QMap<QString, QString> *values*, QMap<QString, int> *onlyinfileviewindex*,  
 bool *withSubfolders*, QStringList *inheritFrom*)

Stores one setting into a profile.

**~SettingsManager** ()

void **doInitialize** ()  
Executes singleton initialization.

void **doShutdown** ()  
Executes singleton shutdown.

void **shutdown** ()  
Shutdown down this singleton.

bool **isAlive** ()  
Returns if this singleton is alive (true until its shutdown begins).

## Signals

void **profilesChanged** ()  
Triggered when the list of profiles or the nodes inside it change.

Can also be triggered when nothing really changed!

## Private Functions

void **applyToFileView** (*sh::ui::FileView* \*list, QMap<QString, QString> settings)  
Applies some settings (as string tuples) to a fileview.

void **applyToMainWindow** (QMap<QString, QString> settings)  
Applies some settings (as string tuples) to main window.

QMap<QString, QString> **\_getSettings** (std::shared\_ptr<const *sh::filesystem::Eurl*> eurl,  
QString profilename, int fileviewindex, bool directNode = true)  
Computes all effective settings (including inheritance and sub-directories) for a given situation (global or eurl-bound; fileview- or mainwindow-bound) and returns it as string tuples.

### Parameters

- eurl: The directory to consider as current (or nullptr for global settings).
- profilename: The current profile.
- fileviewindex: The fileview index (or -1 for mainwindow-bound settings).
- directNode: If the given directory is to be considered as the real current directory (not e.g. a parent).

QList<*\_SettingsFinderStructure*> **\_getSettings\_flat** (std::shared\_ptr<const *sh::filesystem::Eurl*> eurl, QString profilename)  
Used internally by *\_getSettings*().

void **invalidateCache** ()  
Invalidates the *\_getSettings* cache.

void **readStoredProfiles** ()  
Reads all profiles (including all setting nodes) from filesystem.

**SettingsManager** ()

## Private Members

```
QMap<QString, QMap<QString, QString>> _getSettings_cache
QMap<QString, std::shared_ptr<sh::settings::Setting>> _settings
QMap<std::shared_ptr<const sh::filesystem::Eurl>, QMap<QString, QList<sh::settings::ProfileNode*>>> _profileNo
QMap<QString, sh::settings::ProfileNode*> _profileNodesById
QList<QString> _profileNames
```

## Friends

```
friend class sh::settings::SettingsRegistration

class _MyHandler : public QXmlDefaultHandler
    Used internally for reading profile node XMLs.
```

## Public Functions

```
_MyHandler (sh::settings::ProfileNode *node, QString filename)

bool startElement (const QString &namespaceURI, const QString &localName,
                  const QString &qName, const QXmlAttributes &atts)

bool endElement (const QString &namespaceURI, const QString &localName, const
                QString &qName)
```

## Private Functions

```
void throwError ()
```

## Private Members

```
bool _stateStarted = true
bool _stateInProfile = false
sh::settings::ProfileNode *_node
QString _filename

class _SettingsFinderStructure
    Used internally by _getSettings().
```

## Public Functions

```
_SettingsFinderStructure (QMap<QString, QString> settings, QMap<QString, int>
                          onlyinfileviewindexes, QStringList inheritFrom, bool with-
                          Subfolders)

_SettingsFinderStructure ()
```

### Public Members

QMap<QString, QString> **\_settings**  
QMap<QString, int> **\_onlyinfileviewindexes**  
QStringList **\_inheritFrom**  
bool **\_withSubfolders**  
int **\_onlyFileviewIndex**

### class SettingsRegistration

*#include <settingsregistration.h>* Used for registering a setting instance.

Constructing a *SettingsRegistration* automatically registers a *sh::settings::Setting*, deleting a one automatically unregisters it.

### Public Functions

**SettingsRegistration** (std::shared\_ptr<*sh::settings::Setting*> *setting*)

Is intended to be directly constructed from everywhere.

**~SettingsRegistration** ()

std::shared\_ptr<*sh::settings::Setting*> **setting** ()

### Private Members

std::shared\_ptr<*sh::settings::Setting*> **\_setting**

### namespace common

Implementations of settings.

Subclasses of *sh::settings::Setting* (and possibly some auxiliary stuff). Everything here could be listed in the ‘Save settings’ dialog.

**class FileDetailsPanelVisible : public *sh::settings::Setting***

*#include <filedetailspanelvisible.h>*

### Public Functions

**FileDetailsPanelVisible** ()

QString **name** ()

Gets the internal name.

QString **description** ()

Gets the description text.

*sh::settings::SettingGroup* **group** ()

Gets the group;.

bool **isAdvancedSetting** ()

Is this an advanced setting?

void **setValue** (QString *value*)

Called from Shallot core when the value was set (for a not-per-fileview setting).



QString **getValue** (*sh::ui::FileView \*filelist*)

Get the currently set value.

bool **isGlobal** ()

Does this setting apply globally or just for a certain subtree of nodes?

bool **isPerFileview** ()

Does this setting apply for each fileview individually or for the complete main window?

QString **valueDescription** (QString *value*)

Gets a human readable description text for a value.

void **setValue** (*sh::ui::FileView\**, QString)

Called from Shallot core when the value was set (for a per-fileview setting).

## Public Static Functions

QString **getGroupDescription** (int *g*)

Low-level function which gets the description text of a group.

```
class FileViewIconListThumbDimension : public sh::settings::Setting
#include <fileviewiconlistthumbdimension.h>
```

## Public Functions

**FileViewIconListThumbDimension** ()

QString **name** ()

Gets the internal name.

QString **description** ()

Gets the description text.

*sh::settings::SettingGroup* **group** ()

Gets the group;.

bool **isAdvancedSetting** ()

Is this an advanced setting?

void **setValue** (*sh::ui::FileView \*filelist*, QString *value*)

Called from Shallot core when the value was set (for a per-fileview setting).

QString **getValue** (*sh::ui::FileView \*filelist*)

Get the currently set value.

bool **isGlobal** ()

Does this setting apply globally or just for a certain subtree of nodes?

bool **isPerFileview** ()

Does this setting apply for each fileview individually or for the complete main window?

QString **valueDescription** (QString *value*)

Gets a human readable description text for a value.

void **setValue** (QString)

Called from Shallot core when the value was set (for a not-per-fileview setting).

### Public Static Functions

QString **getGroupDescription** (int g)  
Low-level function which gets the description text of a group.

```
class FileViewPath : public sh::settings::Setting
#include <fileviewpath.h>
```

### Public Functions

**FileViewPath** ()

QString **name** ()  
Gets the internal name.

QString **description** ()  
Gets the description text.

*sh::settings::SettingGroup* **group** ()  
Gets the group;.

bool **isAdvancedSetting** ()  
Is this an advanced setting?

void **setValue** (*sh::ui::FileView* \*fileview, QString value)  
Called from Shallot core when the value was set (for a per-fileview setting).

QString **getValue** (*sh::ui::FileView* \*filelist)  
Get the currently set value.

bool **isGlobal** ()  
Does this setting apply globally or just for a certain subtree of nodes?

bool **isPerFileview** ()  
Does this setting apply for each fileview individually or for the complete main window?

void **setValue** (QString)  
Called from Shallot core when the value was set (for a not-per-fileview setting).

QString **valueDescription** (QString value)  
Gets a human readable description text for a value.

### Public Static Functions

QString **getGroupDescription** (int g)  
Low-level function which gets the description text of a group.

```
class FileViewViewmode : public sh::settings::Setting
#include <fileviewviewmode.h>
```

## Public Functions

**FileViewViewmode** ()

QString **name** ()

Gets the internal name.

QString **description** ()

Gets the description text.

*sh::settings::SettingGroup* **group** ()

Gets the group;.

bool **isAdvancedSetting** ()

Is this an advanced setting?

void **setValue** (*sh::ui::FileView* \*fileview, QString value)

Called from Shallot core when the value was set (for a per-fileview setting).

QString **getValue** (*sh::ui::FileView* \*filelist)

Get the currently set value.

bool **isGlobal** ()

Does this setting apply globally or just for a certain subtree of nodes?

bool **isPerFileview** ()

Does this setting apply for each fileview individually or for the complete main window?

QString **valueDescription** (QString value)

Gets a human readable description text for a value.

void **setValue** (QString)

Called from Shallot core when the value was set (for a not-per-fileview setting).

## Public Static Functions

QString **getGroupDescription** (int g)

Low-level function which gets the description text of a group.

```
class JumpbarMode : public sh::settings::Setting
#include <jumpbarmode.h>
```

## Public Functions

**JumpbarMode** ()

QString **name** ()

Gets the internal name.

QString **description** ()

Gets the description text.

*sh::settings::SettingGroup* **group** ()

Gets the group;.

bool **isAdvancedSetting** ()

Is this an advanced setting?

void **setValue** (QString value)

Called from Shallot core when the value was set (for a not-per-fileview setting).

QString **getValue** (*sh::ui::FileView* \*filelist)

Get the currently set value.

bool **isGlobal** ()

Does this setting apply globally or just for a certain subtree of nodes?

bool **isPerFileview** ()

Does this setting apply for each fileview individually or for the complete main window?

QString **valueDescription** (QString value)

Gets a human readable description text for a value.

void **setValue** (*sh::ui::FileView*\*, QString)

Called from Shallot core when the value was set (for a per-fileview setting).

## Public Static Functions

QString **getGroupDescription** (int g)

Low-level function which gets the description text of a group.

```
class NumberOfFilePanels : public sh::settings::Setting  
#include <numberoffilepanels.h>
```

## Public Functions

**NumberOfFilePanels** ()

QString **name** ()

Gets the internal name.

QString **description** ()

Gets the description text.

*sh::settings::SettingGroup* **group** ()

Gets the group;.

bool **isAdvancedSetting** ()

Is this an advanced setting?

void **setValue** (QString value)

Called from Shallot core when the value was set (for a not-per-fileview setting).

QString **getValue** (*sh::ui::FileView* \*filelist)

Get the currently set value.

bool **isGlobal** ()

Does this setting apply globally or just for a certain subtree of nodes?

bool **isPerFileview** ()

Does this setting apply for each fileview individually or for the complete main window?

QString **valueDescription** (QString value)

Gets a human readable description text for a value.

void **setValue** (*sh::ui::FileView*\*, QString)

Called from Shallot core when the value was set (for a per-fileview setting).

## Public Static Functions

QString **getGroupDescription** (int *g*)  
 Low-level function which gets the description text of a group.

```
class ShowHiddenFiles : public sh::settings::Setting
#include <showhiddenfiles.h>
```

## Public Functions

ShowHiddenFiles ()

QString **name** ()  
 Gets the internal name.

QString **description** ()  
 Gets the description text.

sh::settings::SettingGroup **group** ()  
 Gets the group;.

bool **isAdvancedSetting** ()  
 Is this an advanced setting?

void **setValue** (sh::ui::FileView \**filelist*, QString *value*)  
 Called from Shallot core when the value was set (for a per-fileview setting).

QString **getValue** (sh::ui::FileView \**filelist*)  
 Get the currently set value.

bool **isGlobal** ()  
 Does this setting apply globally or just for a certain subtree of nodes?

bool **isPerFileview** ()  
 Does this setting apply for each fileview individually or for the complete main window?

QString **valueDescription** (QString *value*)  
 Gets a human readable description text for a value.

void **setValue** (QString)  
 Called from Shallot core when the value was set (for a not-per-fileview setting).

## Public Static Functions

QString **getGroupDescription** (int *g*)  
 Low-level function which gets the description text of a group.

```
class ShowTree : public sh::settings::Setting
#include <showtree.h>
```

## Public Functions

**ShowTree** ()

QString **name** ()

Gets the internal name.

QString **description** ()

Gets the description text.

*sh::settings::SettingGroup* **group** ()

Gets the group;.

bool **isAdvancedSetting** ()

Is this an advanced setting?

void **setValue** (QString *value*)

Called from Shallot core when the value was set (for a not-per-fileview setting).

QString **getValue** (*sh::ui::FileView* \**filelist*)

Get the currently set value.

bool **isGlobal** ()

Does this setting apply globally or just for a certain subtree of nodes?

bool **isPerFileview** ()

Does this setting apply for each fileview individually or for the complete main window?

QString **valueDescription** (QString *value*)

Gets a human readable description text for a value.

void **setValue** (*sh::ui::FileView*\*, QString)

Called from Shallot core when the value was set (for a per-fileview setting).

## Public Static Functions

QString **getGroupDescription** (int *g*)

Low-level function which gets the description text of a group.

```
class SizeFormattingMode : public sh::settings::Setting  
    #include <sizeformattingmode.h>
```

## Public Functions

**SizeFormattingMode** ()

QString **name** ()

Gets the internal name.

QString **description** ()

Gets the description text.

*sh::settings::SettingGroup* **group** ()

Gets the group;.

bool **isAdvancedSetting** ()

Is this an advanced setting?

void **setValue** (*sh::ui::FileView* \**filelist*, QString *value*)

Called from Shallot core when the value was set (for a per-fileview setting).

QString **getValue** (*sh::ui::FileView \*filelist*)

Get the currently set value.

bool **isGlobal** ()

Does this setting apply globally or just for a certain subtree of nodes?

bool **isPerFileview** ()

Does this setting apply for each fileview individually or for the complete main window?

QString **valueDescription** (QString *value*)

Gets a human readable description text for a value.

void **setValue** (QString)

Called from Shallot core when the value was set (for a not-per-fileview setting).

## Public Static Functions

QString **getGroupDescription** (int *g*)

Low-level function which gets the description text of a group.

```
class StickyTreeview: public sh::settings::Setting
#include <stickytreview.h>
```

## Public Functions

**StickyTreeview** ()

QString **name** ()

Gets the internal name.

QString **description** ()

Gets the description text.

*sh::settings::SettingGroup* **group** ()

Gets the group;.

bool **isAdvancedSetting** ()

Is this an advanced setting?

void **setValue** (QString *value*)

Called from Shallot core when the value was set (for a not-per-fileview setting).

QString **getValue** (*sh::ui::FileView \*filelist*)

Get the currently set value.

bool **isGlobal** ()

Does this setting apply globally or just for a certain subtree of nodes?

bool **isPerFileview** ()

Does this setting apply for each fileview individually or for the complete main window?

QString **valueDescription** (QString *value*)

Gets a human readable description text for a value.

void **setValue** (*sh::ui::FileView\**, QString)

Called from Shallot core when the value was set (for a per-fileview setting).

### Public Static Functions

QString **getGroupDescription** (int *g*)  
Low-level function which gets the description text of a group.

```
class WindowTitle : public sh::settings::Setting
#include <windowtitle.h>
```

### Public Functions

WindowTitle ()

QString **name** ()  
Gets the internal name.

QString **description** ()  
Gets the description text.

sh::settings::SettingGroup **group** ()  
Gets the group;.

bool **isAdvancedSetting** ()  
Is this an advanced setting?

void **setValue** (QString *value*)  
Called from Shallot core when the value was set (for a not-per-fileview setting).

QString **getValue** (sh::ui::FileView \**filelist*)  
Get the currently set value.

bool **isGlobal** ()  
Does this setting apply globally or just for a certain subtree of nodes?

bool **isPerFileview** ()  
Does this setting apply for each fileview individually or for the complete main window?

void **setValue** (sh::ui::FileView\*, QString)  
Called from Shallot core when the value was set (for a per-fileview setting).

QString **valueDescription** (QString *value*)  
Gets a human readable description text for a value.

### Public Static Functions

QString **getGroupDescription** (int *g*)  
Low-level function which gets the description text of a group.

## 10.2.20 Namespace sh::settings::common

namespace *sh::settings::common*  
Implementations of settings.

Subclasses of *sh::settings::Setting* (and possibly some auxiliary stuff). Everything here could be listed in the ‘Save settings’ dialog.

```
class FileDetailsPanelVisible : public sh::settings::Setting
#include <filedetailspanelvisible.h>
```



## Public Functions

**FileDetailsPanelVisible** ()

QString **name** ()

Gets the internal name.

QString **description** ()

Gets the description text.

*sh::settings::SettingGroup* **group** ()

Gets the group;.

bool **isAdvancedSetting** ()

Is this an advanced setting?

void **setValue** (QString *value*)

Called from Shallot core when the value was set (for a not-per-fileview setting).

QString **getValue** (*sh::ui::FileView \*filelist*)

Get the currently set value.

bool **isGlobal** ()

Does this setting apply globally or just for a certain subtree of nodes?

bool **isPerFileview** ()

Does this setting apply for each fileview individually or for the complete main window?

QString **valueDescription** (QString *value*)

Gets a human readable description text for a value.

void **setValue** (*sh::ui::FileView\**, QString)

Called from Shallot core when the value was set (for a per-fileview setting).

## Public Static Functions

QString **getGroupDescription** (int *g*)

Low-level function which gets the description text of a group.

```
class FileViewIconListThumbDimension : public sh::settings::Setting
#include <fileviewiconlistthumbdimension.h>
```

## Public Functions

**FileViewIconListThumbDimension** ()

QString **name** ()

Gets the internal name.

QString **description** ()

Gets the description text.

*sh::settings::SettingGroup* **group** ()

Gets the group;.

bool **isAdvancedSetting** ()

Is this an advanced setting?

void **setValue** (*sh::ui::FileView \*filelist*, QString *value*)

Called from Shallot core when the value was set (for a per-fileview setting).

QString **getValue** (*sh::ui::FileView \*filelist*)

Get the currently set value.

bool **isGlobal** ()

Does this setting apply globally or just for a certain subtree of nodes?

bool **isPerFileview** ()

Does this setting apply for each fileview individually or for the complete main window?

QString **valueDescription** (QString *value*)

Gets a human readable description text for a value.

void **setValue** (QString)

Called from Shallot core when the value was set (for a not-per-fileview setting).

## Public Static Functions

QString **getGroupDescription** (int *g*)

Low-level function which gets the description text of a group.

```
class FileViewPath: public sh::settings::Setting
#include <fileviewpath.h>
```

## Public Functions

**FileViewPath** ()

QString **name** ()

Gets the internal name.

QString **description** ()

Gets the description text.

*sh::settings::SettingGroup* **group** ()

Gets the group;.

bool **isAdvancedSetting** ()

Is this an advanced setting?

void **setValue** (*sh::ui::FileView \*fileview*, QString *value*)

Called from Shallot core when the value was set (for a per-fileview setting).

QString **getValue** (*sh::ui::FileView \*filelist*)

Get the currently set value.

bool **isGlobal** ()

Does this setting apply globally or just for a certain subtree of nodes?

bool **isPerFileview** ()

Does this setting apply for each fileview individually or for the complete main window?

void **setValue** (QString)

Called from Shallot core when the value was set (for a not-per-fileview setting).

QString **valueDescription** (QString *value*)

Gets a human readable description text for a value.

## Public Static Functions

QString **getGroupDescription** (int g)  
 Low-level function which gets the description text of a group.

```
class FileViewViewmode : public sh::settings::Setting
#include <fileviewviewmode.h>
```

## Public Functions

**FileViewViewmode** ()

QString **name** ()  
 Gets the internal name.

QString **description** ()  
 Gets the description text.

sh::settings::SettingGroup **group** ()  
 Gets the group;.

bool **isAdvancedSetting** ()  
 Is this an advanced setting?

void **setValue** (sh::ui::FileView \*fileview, QString value)  
 Called from Shallot core when the value was set (for a per-fileview setting).

QString **getValue** (sh::ui::FileView \*filelist)  
 Get the currently set value.

bool **isGlobal** ()  
 Does this setting apply globally or just for a certain subtree of nodes?

bool **isPerFileview** ()  
 Does this setting apply for each fileview individually or for the complete main window?

QString **valueDescription** (QString value)  
 Gets a human readable description text for a value.

void **setValue** (QString)  
 Called from Shallot core when the value was set (for a not-per-fileview setting).

## Public Static Functions

QString **getGroupDescription** (int g)  
 Low-level function which gets the description text of a group.

```
class JumpbarMode : public sh::settings::Setting
#include <jumpbarmode.h>
```

## Public Functions

**JumpbarMode** ()

QString **name** ()

Gets the internal name.

QString **description** ()

Gets the description text.

*sh::settings::SettingGroup* **group** ()

Gets the group;.

bool **isAdvancedSetting** ()

Is this an advanced setting?

void **setValue** (QString *value*)

Called from Shallot core when the value was set (for a not-per-fileview setting).

QString **getValue** (*sh::ui::FileView* \**filelist*)

Get the currently set value.

bool **isGlobal** ()

Does this setting apply globally or just for a certain subtree of nodes?

bool **isPerFileview** ()

Does this setting apply for each fileview individually or for the complete main window?

QString **valueDescription** (QString *value*)

Gets a human readable description text for a value.

void **setValue** (*sh::ui::FileView*\*, QString)

Called from Shallot core when the value was set (for a per-fileview setting).

## Public Static Functions

QString **getGroupDescription** (int *g*)

Low-level function which gets the description text of a group.

```
class NumberOfFilePanels : public sh::settings::Setting  
    #include <numberoffilepanels.h>
```

## Public Functions

**NumberOfFilePanels** ()

QString **name** ()

Gets the internal name.

QString **description** ()

Gets the description text.

*sh::settings::SettingGroup* **group** ()

Gets the group;.

bool **isAdvancedSetting** ()

Is this an advanced setting?

void **setValue** (QString *value*)

Called from Shallot core when the value was set (for a not-per-fileview setting).

QString **getValue** (*sh::ui::FileView \*filelist*)

Get the currently set value.

bool **isGlobal** ()

Does this setting apply globally or just for a certain subtree of nodes?

bool **isPerFileview** ()

Does this setting apply for each fileview individually or for the complete main window?

QString **valueDescription** (QString *value*)

Gets a human readable description text for a value.

void **setValue** (*sh::ui::FileView\**, QString)

Called from Shallot core when the value was set (for a per-fileview setting).

## Public Static Functions

QString **getGroupDescription** (int *g*)

Low-level function which gets the description text of a group.

```
class ShowHiddenFiles : public sh::settings::Setting
#include <showhiddenfiles.h>
```

## Public Functions

**ShowHiddenFiles** ()

QString **name** ()

Gets the internal name.

QString **description** ()

Gets the description text.

*sh::settings::SettingGroup* **group** ()

Gets the group;.

bool **isAdvancedSetting** ()

Is this an advanced setting?

void **setValue** (*sh::ui::FileView \*filelist*, QString *value*)

Called from Shallot core when the value was set (for a per-fileview setting).

QString **getValue** (*sh::ui::FileView \*filelist*)

Get the currently set value.

bool **isGlobal** ()

Does this setting apply globally or just for a certain subtree of nodes?

bool **isPerFileview** ()

Does this setting apply for each fileview individually or for the complete main window?

QString **valueDescription** (QString *value*)

Gets a human readable description text for a value.

void **setValue** (QString)

Called from Shallot core when the value was set (for a not-per-fileview setting).

### Public Static Functions

QString **getGroupDescription** (int g)  
Low-level function which gets the description text of a group.

```
class ShowTree : public sh::settings::Setting
#include <showtree.h>
```

### Public Functions

**ShowTree** ()

QString **name** ()  
Gets the internal name.

QString **description** ()  
Gets the description text.

*sh::settings::SettingGroup* **group** ()  
Gets the group;.

bool **isAdvancedSetting** ()  
Is this an advanced setting?

void **setValue** (QString *value*)  
Called from Shallot core when the value was set (for a not-per-fileview setting).

QString **getValue** (*sh::ui::FileView* \**filelist*)  
Get the currently set value.

bool **isGlobal** ()  
Does this setting apply globally or just for a certain subtree of nodes?

bool **isPerFileview** ()  
Does this setting apply for each fileview individually or for the complete main window?

QString **valueDescription** (QString *value*)  
Gets a human readable description text for a value.

void **setValue** (*sh::ui::FileView*\*, QString)  
Called from Shallot core when the value was set (for a per-fileview setting).

### Public Static Functions

QString **getGroupDescription** (int g)  
Low-level function which gets the description text of a group.

```
class SizeFormattingMode : public sh::settings::Setting
#include <sizeformattingmode.h>
```

## Public Functions

**SizeFormattingMode** ()

QString **name** ()

Gets the internal name.

QString **description** ()

Gets the description text.

*sh::settings::SettingGroup* **group** ()

Gets the group;.

bool **isAdvancedSetting** ()

Is this an advanced setting?

void **setValue** (*sh::ui::FileView* \*filelist, QString value)

Called from Shallot core when the value was set (for a per-fileview setting).

QString **getValue** (*sh::ui::FileView* \*filelist)

Get the currently set value.

bool **isGlobal** ()

Does this setting apply globally or just for a certain subtree of nodes?

bool **isPerFileview** ()

Does this setting apply for each fileview individually or for the complete main window?

QString **valueDescription** (QString value)

Gets a human readable description text for a value.

void **setValue** (QString)

Called from Shallot core when the value was set (for a not-per-fileview setting).

## Public Static Functions

QString **getGroupDescription** (int g)

Low-level function which gets the description text of a group.

```
class StickyTreeview: public sh::settings::Setting
#include <stickytreeview.h>
```

## Public Functions

**StickyTreeview** ()

QString **name** ()

Gets the internal name.

QString **description** ()

Gets the description text.

*sh::settings::SettingGroup* **group** ()

Gets the group;.

bool **isAdvancedSetting** ()

Is this an advanced setting?

void **setValue** (QString value)

Called from Shallot core when the value was set (for a not-per-fileview setting).

QString **getValue** (*sh::ui::FileView \*filelist*)

Get the currently set value.

bool **isGlobal** ()

Does this setting apply globally or just for a certain subtree of nodes?

bool **isPerFileview** ()

Does this setting apply for each fileview individually or for the complete main window?

QString **valueDescription** (QString *value*)

Gets a human readable description text for a value.

void **setValue** (*sh::ui::FileView\**, QString)

Called from Shallot core when the value was set (for a per-fileview setting).

## Public Static Functions

QString **getGroupDescription** (int *g*)

Low-level function which gets the description text of a group.

```
class WindowTitle : public sh::settings::Setting
#include <windowtitle.h>
```

## Public Functions

**WindowTitle** ()

QString **name** ()

Gets the internal name.

QString **description** ()

Gets the description text.

*sh::settings::SettingGroup* **group** ()

Gets the group;.

bool **isAdvancedSetting** ()

Is this an advanced setting?

void **setValue** (QString *value*)

Called from Shallot core when the value was set (for a not-per-fileview setting).

QString **getValue** (*sh::ui::FileView \*filelist*)

Get the currently set value.

bool **isGlobal** ()

Does this setting apply globally or just for a certain subtree of nodes?

bool **isPerFileview** ()

Does this setting apply for each fileview individually or for the complete main window?

void **setValue** (*sh::ui::FileView\**, QString)

Called from Shallot core when the value was set (for a per-fileview setting).

QString **valueDescription** (QString *value*)

Gets a human readable description text for a value.



### Public Static Functions

QString **getGroupDescription** (int *g*)

Low-level function which gets the description text of a group.

## 10.2.21 Namespace `sh::tools`

namespace `sh::tools`

Auxiliary stuff.

class **AtomicCounter**

*#include <atomiccounter.h>* Internal tool for reference counting.

### Public Functions

**AtomicCounter** ()

int **value** ()

std::shared\_ptr<*AtomicCounter::Increment*> **increment** ()

int **doIncValue** ()

int **doDecValue** ()

### Private Members

int **\_value** = 0

QMutex **\_mutex**

class **Increment**

*#include <atomiccounter.h>*

### Public Functions

**Increment** (*AtomicCounter \*ac*)

**~Increment** ()

### Private Members

*AtomicCounter \*\_ac*

## Friends

```
friend class AtomicCounter

class Benchmarking : public QObject, public sh::base::Singleton
    #include <benchmarking.h> Tools for performance measurements.
```

## Public Types

```
typedef QPair<QString, qint64> BenchmarkFrame
    A pair of a benchmark keyname and a duration value.
```

## Public Functions

```
Benchmarking ()

void shutdown ()
    Shutdown down this singleton.

bool isAlive ()
    Returns if this singleton is alive (true until its shutdown begins).

class Bookmark
    #include <bookmarkmanager.h> A bookmark.
```

## Public Functions

```
Bookmark (QString id, QStringList folder, QString label, std::shared_ptr<const
    sh::filesystem::Eurl> eurl, QString tags)
    Constructed only by the infrastructure and made available otherwise.

QString id ()
    The bookmark id (used in some methods of sh::tools::BookmarkManager).

QList<QString> folder ()
    The bookmark folder (the user interface calls it 'Collections').

QString label ()
    The bookmark label.

std::shared_ptr<const sh::filesystem::Eurl> eurl ()
    The location this bookmark points to.

QString tags ()
    Internal information for bookkeeping.

    Can be used by external code for keeping track of dynamically created bookmarks.
```

## Private Members

QString **\_id**  
 QStringList **\_folder**  
 QString **\_label**  
 std::shared\_ptr<const *sh::filesystem::Eurl*> **\_eurl**  
 QString **\_tags**

**class BookmarkManager** : public QObject, public *sh::base::Singleton*  
*#include <bookmarkmanager.h>* The bookmark manager.

Use it for getting or modifying bookmarks.

## Public Functions

QList<std::shared\_ptr<*sh::tools::Bookmark*>> **getBookmarks** ()  
 Returns a list of all stored bookmarks.

bool **hasBookmarks** ()  
 Checks if any bookmarks exist.

QString **addBookmark** (QList<QString> *folder*, QString *label*, std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*, QString *tags* = QString())  
 Adds a new bookmark.

**Return** The id of the new bookmark.

void **removeBookmark** (QString *id*)  
 Removes a bookmarks.

void **changeBookmark** (QString *id*, QString *label*, std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*)  
 Changes data of a bookmark.

void **changeBookmarkTags** (QString *id*, QString *tags*)  
 Changes the *tags* of a bookmark.

void **moveBookmarkUp** (QString *id*)  
 Moves a bookmark up within its folder.

void **moveBookmarkDown** (QString *id*)  
 Moves a bookmark down within its folder.

void **moveBookmarkToFolder** (QString *id*, QList<QString> *folder*)  
 Changes to *folder* of a bookmark.

void **doInitialize** ()  
 Executes singleton initialization.

void **doShutdown** ()  
 Executes singleton shutdown.

void **shutdown** ()  
 Shutdown down this singleton.

bool **isAlive** ()  
 Returns if this singleton is alive (true until its shutdown begins).

## Signals

void **changed** ()

## Private Functions

**BookmarkManager** ()

void **moveBookmark** (QString *id*, int *direction*)

void **\_changeBookmark** (QString *id*, std::function<QString> std::shared\_ptr<Bookmark>  
> *label*, std::function<std::shared\_ptr<const sh::filesystem::Eurl>std::shared\_ptr<Bookmark>>  
*eurl*, std::function<QList<QString>std::shared\_ptr<Bookmark>> *folder*,  
std::function<QStringstd::shared\_ptr<Bookmark>> *tags*)

void **readBookmarks** ()

void **writeBookmarks** ()

## Private Members

std::shared\_ptr<sh::configuration::ConfigurationValue> **cfgvalBookmarks**

QList<std::shared\_ptr<Bookmark>> **\_bookmarks**

QMutex **\_bookmarksmutex**

**class DataExchange** : public QObject, public sh::base::Singleton  
*#include <dataexchange.h>* Clipboard and DnD related tools.

## Public Types

**enum DataExchangeType**

A kind of data exchange movement.

*Values:*

**enumerator COPY**

**enumerator MOVE**

## Public Functions

bool **containsFileEntries** (const QMimeData\*)

Checks if a QMimeData contains any file entries we understand.

QMimeData \***getMimeDataFromFilesystemNodes** (QList<std::shared\_ptr<sh::filesystem::FilesystemNode>>, *DataExchangeType type*)

Returns a new QMimeData for a list of filesystem nodes and an exchange type.

std::shared\_ptr<sh::actions::AbstractActionItem> **createCopyAction** (const QMime-  
Data \**src*,  
std::shared\_ptr<const  
sh::filesystem::Eurl>  
*dest*)

Returns a copy action for a source by QMimeData and a destination eurl.

```
std::shared_ptr<sh::actions::AbstractActionItem> createMoveAction (const      QMime-
                                                                    Data          *src,
                                                                    std::shared_ptr<const
sh::filesystem::Eurl>
                                                                    dest)
```

Returns a move action for a source by QMimeData and a destination eurl.

```
std::shared_ptr<sh::actions::AbstractActionItem> createPasteAction (const      QMime-
                                                                    Data          *src,
                                                                    std::shared_ptr<const
sh::filesystem::Eurl>
                                                                    dest)
```

Returns a paste action for a source by QMimeData and a destination eurl (copy or move by QMimeData).

```
QList<std::shared_ptr<const sh::filesystem::Eurl>> getSources (const      QMimeData      *src,
                                                                    bool *isCut = 0)
```

Returns a list of eurls (and if it is a cut/move exchange) from the QMimeData.

```
void doInitialize ()
    Executes singleton initialization.
```

```
void doShutdown ()
    Executes singleton shutdown.
```

```
void shutdown ()
    Shutdown down this singleton.
```

```
bool isAlive ()
    Returns if this singleton is alive (true until its shutdown begins).
```

### Public Static Attributes

```
QString FilelistTypeShallot = "x-special/shallot-copied-files"
```

```
QString FilelistTypeGnome = "x-special/gnome-copied-files"
```

```
QString FilelistTypeUrulist = "text/uri-list"
```

```
QString FilelistTypePlaintext = "text/plain"
```

### Private Functions

```
DataExchange ()
```

### Private Static Attributes

```
QString linebreak = "\n"
```

```
template<class T>
```

```
class HistoryTracker
```

```
#include <historytracker.h> A data structure for tracking a history of data.
```

It is a templated container It is typically used for tracking the directory history (used by *sh::actions::common::ActionHistoryNavigate*, et al).

### Public Functions

```
HistoryTracker ()  
void visitValue (T v)  
int count ()  
QList<HistoryEntry> forwardList ()  
QList<HistoryEntry> backwardList ()  
void revisitValue (int idx)
```

### Public Members

```
const int HISTORY_SIZE = 10
```

### Private Members

```
QList<T> _items  
int _current = -1  
class HistoryEntry  
    #include <historytracker.h>
```

### Public Functions

```
HistoryEntry (int i, T v)  
int index ()  
T value ()
```

### Private Members

```
int _index  
T _value
```

### class Jsonable

*#include <misc.h>* An interface for objects which can return a json representation as QJsonObject.

Subclassed by *sh::ui::web::WebDialog*, *sh::ui::web::WebFilePropertyDialog::WebFilePropertyDialogTabActionsView*,  
*sh::ui::web::WebFilePropertyDialog::WebFilePropertyDialogTabIconTextBannerView*,  
*sh::ui::web::WebFilePropertyDialog::WebFilePropertyDialogTabTableView*,  
*sh::ui::web::WebFilePropertyDialog::WebFilePropertyDialogTabTextView*

## Public Functions

**QObject toJson () = 0**

Returns the json representation as QObject.

**class LocalFile : public QFile**

*#include <datastream.h>* A QFile (and a QIODevice) for local file access.

It also stores the information whether this is a temporary location or the permanent one.

## Public Functions

**LocalFile** (QString &file, bool isTemp = false)

bool **isTemp** ()

## Private Members

bool **\_istemp**

**class LocalFilesystemWatcher : public QObject, public *sh::base::Singleton***

*#include <localfilesystemwatcher.h>* Used for watching parts of the local filesystem.

Depending on the compile flags and your system, this functionality might not be available. If so, the methods are no-ops.

## Public Functions

qint64 **addFile** (QString path)

Watches one more file.

See also *removeFile()*.

void **removeFile** (qint64 id)

Stops watching a file (by the return value of *addFile()*).

qint64 **addDirectory** (QString path)

Watches one more directory.

See also *removeDirectory()*.

void **removeDirectory** (qint64 id)

Stops watching a directory (by the return value of *addDirectory()*).

**~LocalFilesystemWatcher** ()

void **doInitialize** ()

Executes singleton initialization.

void **doShutdown** ()

Executes singleton shutdown.

void **shutdown** ()

Shutdown down this singleton.

bool **isAlive** ()

Returns if this singleton is alive (true until its shutdown begins).

## Signals

```
void elementModified (QString name, QList<qint64> ids)  
void elementDeleted (QString name, QList<qint64> ids)  
void elementCreated (QString name, QList<qint64> ids)
```

## Private Functions

```
LocalFilesystemWatcher ()  
void emitElementModified (QString name, QList<qint64> ids)  
void emitElementDeleted (QString name, QList<qint64> ids)  
void emitElementCreated (QString name, QList<qint64> ids)
```

## Private Members

```
InotifyThread *inotifyThread  
QMutex inotifymutex  
QHash<int, QList<qint64>> inotifywd2ids  
QHash<qint64, int> id2inotifywd  
QHash<qint64, QString> id2path  
qint64 nextEid = 0  
QMutex inotifythreadmutex  
QWaitCondition inotifythreadmutexwait
```

## Friends

```
friend class InotifyThread  
class InotifyThread : public QThread
```

## Public Functions

```
InotifyThread (LocalFilesystemWatcher *watcher)
```

## Friends

```
friend class LocalFilesystemWatcher  
class LocalFilesystemWatcherConnector : public QObject, public sh::base::Singleton  
    #include <localfilesystemwatcherconnector.h> Observes the list of file views via VisibleViews and controls  
    LocalFilesystemWatcher with that for observing the filesystem for changes.
```



## Public Functions

void **doInitialize** ()  
 Executes singleton initialization.

void **doShutdown** ()  
 Executes singleton shutdown.

void **shutdown** ()  
 Shutdown down this singleton.

bool **isAlive** ()  
 Returns if this singleton is alive (true until its shutdown begins).

## Private Functions

**LocalFilesystemWatcherConnector** ()  
 QList<QPair<std::shared\_ptr<*sh::filesystem::FilesystemNode*>, *sh::filesystemhandlers::LocalFilesystemHandler*\*>> **getAl**

## Private Members

QHash<std::shared\_ptr<const *sh::filesystem::Eurl*>, QList<qint64>> **dir2watcherids**  
 QHash<qint64, std::shared\_ptr<const *sh::filesystem::Eurl*>> **watcherid2dir**

**class Misc**  
*#include <misc.h>*

## Public Static Functions

QByteArray **iconToPngByteArray** (QIcon *icon*, int *sizeInPt*)  
 Computes a png representation for an icon.

QByteArray **iconToBase64SrcEncoding** (QIcon *icon*, int *sizeInPt*)  
 Computes a html-compatible 'data:image/png;base64,...' png representation for an icon.

QByteArray **hash** (QStringList *data*)  
 Hashes string data in a cryptographical way (with a salt).  
 Use with *compareHash()*.

QByteArray **hashUnsalted** (QStringList *data*)  
 Hashes string data in a cryptographical way without a salt.  
 Use with *compareHash()* or just compare strings.

bool **compareHash** (QByteArray *hash*, QStringList *data*)  
 Check if a hash is matching to given string data.

QByteArray **qjsonToJson** (QJsonObject *o*)  
 Returns a json string for a QJsonObject, QJsonValue or QJsonArray.

QByteArray **qjsonToJson** (QJsonValue *o*)

QByteArray **qjsonToJson** (QJsonArray *a*)

QByteArray **qmapToJson** (QVariantMap *m*)  
 Returns a json string for a QVariantMap.

QByteArray **jsonableToJson** (*Jsonable* \*a)

Returns a json string from a *Jsonable*.

QByteArray **randomBytes** (int *length* = 32)

Returns a random byte array.

QByteArray **generateUniqueHash** ()

Returns a (mostly) random generated unique hash.

void **makeHttpRequest** (QUrl *url*, QByteArray \**responseBody* = nullptr, QString \**mimeType* =  
nullptr, int \**httpStatus* = nullptr)

Makes an http request (as a client).

*httpStatus* can return 0 for network errors and similar.

## Private Functions

**Misc** ()

## Private Static Functions

QByteArray **\_hash** (QStringList *data*, bool *salted*, QByteArray *\_withSalt*)

## Private Static Attributes

QByteArray **\_\_jundefined**

QNetworkAccessManager **\_qnetwork**

**class OperationsCache** : public QObject, public *sh::base::Singleton*

*#include <operationscache.h>* A cached factory for readonly *sh::filesystem::Operation* instances.

You can use it for fetching an *sh::filesystem::Operation* for a certain *sh::filesystem::Eurl*. Two restrictions apply:

- Data might be slightly outdated.
- Just for read-only access.

## Public Functions

std::shared\_ptr<*sh::filesystem::Operation*> **getOperationForContainer** (std::shared\_ptr<const  
*sh::filesystem::Eurl*>  
*container*)

Returns a cached *sh::filesystem::Operation* for a container (by eurl).

void **doInitialize** ()

Executes singleton initialization.

void **doShutdown** ()

Executes singleton shutdown.

void **shutdown** ()

Shutdown down this singleton.

bool **isAlive** ()

Returns if this singleton is alive (true until its shutdown begins).

## Private Functions

**OperationsCache** ()

## Private Members

QHash<std::shared\_ptr<const *sh::filesystem::Eurl*>, std::shared\_ptr<*sh::filesystem::Operation*>> **cache**  
 QMutex **cachemutex**

**class ReadDataDevice** : public QIODevice  
*#include <datastream.h>* A QIODevice implementation, acting as an abstract base class for other sub-classes.

The provided interface is not as generic but easier than the original one for many easier tasks.

Subclassed by *sh::scripting::api::ApiReadDataDevice*

## Public Functions

**ReadDataDevice** ()

QByteArray **getdata** () = 0

This method returns the next available chunk of data. It may return empty arrays whenever temporarily no data is available. Returning a null array (with `QByteArray::isNull() == true`) marks the end of the stream. .

## Private Members

QByteArray **current**

int **currentlen** = 0

int **currentconsumed** = 0

**class ThumbnailManager** : public QObject, public *sh::base::Singleton*  
*#include <thumbnailmanager.h>* Creates thumbnail images for filesystem nodes.

This is a cached source.

## Public Functions

void **requestThumbnail** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> *node*, int *width*, int *height*, std::function<void> *QIcon*  
 > *callback* = 0) Requests a thumbnail in a given size for a given node.

### Parameters

- *callback*: Called when the thumbnail is ready.

bool **getThumbnail** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> *node*, int *width*, int *height*,  
 QIcon \**result*, bool \**outdated* = 0, bool \**refreshRequested* = 0)  
 Returns a thumbnail in a given size for a given node from the current cache.

void **invalidateCache** ()  
 Invalidates the thumbnail cache.

void **addThumbnailProvider** (int *index*, std::shared\_ptr<*ThumbnailProvider*> *provider*)  
 Adds a thumbnail provider.

void **doInitialize** ()  
Executes singleton initialization.

void **shutdown** ()  
Shutdown down this singleton.

bool **isAlive** ()  
Returns if this singleton is alive (true until its shutdown begins).

## Signals

void **thumbnailAvailable** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> node)  
Emitted when a new thumbnail is available.

## Public Static Attributes

const int **REGISTER\_THUMBNAILPROVIDER\_INDEX\_CORE** = 1000000  
Base value for display indexes of core (i.e. very important) thumbnail providers.  
Used in ThumbnailProvider::addThumbnailProvider.

const int **REGISTER\_THUMBNAILPROVIDER\_INDEX\_NORMAL** = 2000000  
Base value for display indexes of thumbnail providers with normal importance.  
Used in ThumbnailProvider::addThumbnailProvider.

const int **REGISTER\_THUMBNAILPROVIDER\_INDEX\_EXOTIC** = 3000000  
Base value for display indexes of exotic thumbnail providers.  
Used in ThumbnailProvider::addThumbnailProvider.

const int **REGISTER\_THUMBNAILPROVIDER\_INDEX\_FALLBACK** = 4000000  
Base value for display indexes of fallback thumbnail providers (those who try to get something when everything else failed).  
Used in ThumbnailProvider::addThumbnailProvider.

## Private Functions

**ThumbnailManager** ()

void **worker** ()

void **\_enforce\_capacity** ()

## Private Members

QHash<*sh::filesystem::FilesystemNode*\*, *Thumbnail*\*> **thumbnails**

QMutex **thumbnailslock**

QList<*ThumbnailRequest*> **requestQueue**

int **maxworkers**

int **runningworkers**

QMutex **workermutex**

int **capacity**

```

qint64 _curr_accessTime = 0
QHash<int, std::shared_ptr<ThumbnailProvider>> _thumbnailProviderMap
QList<std::shared_ptr<ThumbnailProvider>> _thumbnailProviders
QMutex _thumbnailProvidersMutex
class Thumbnail

```

### Public Functions

```

Thumbnail (QIcon icon, qint64 validUntil, qint64 accessTime,
           std::weak_ptr<sh::filesystem::FilesystemNode> node, int width, int height)

```

### Public Members

```

QIcon icon
qint64 validUntil
qint64 accessTime
bool refreshRequested
std::weak_ptr<sh::filesystem::FilesystemNode> node
int width
int height
class ThumbnailRequest

```

### Public Functions

```

ThumbnailRequest (std::weak_ptr<sh::filesystem::FilesystemNode> node, int width, int
                  height, QList<std::function<void>()> QIcon
                  >> callbacks)
bool operator== (ThumbnailRequest const &b)

```

### Public Members

```

std::weak_ptr<sh::filesystem::FilesystemNode> node
int width
int height
QList<std::function<void>()> QIcon> callbacks
class ThumbnailSortStruct

```

## Public Functions

**ThumbnailSortStruct** (*sh::filesystem::FileSystemNode \*node*, qint64 *accessTime*)

## Public Members

*sh::filesystem::FileSystemNode \*node*

qint64 **accessTime**

**class ThumbnailProvider**

*#include <thumbnailmanager.h>* Subclassed by *sh::scripting::api::ApiThumbnailProvider*,  
*sh::tools::thumbnailproviders::DefaultImageThumbnailProvider*, *sh::tools::thumbnailproviders::FfmpegVideoThumbnailP*,  
*sh::tools::thumbnailproviders::ImageMagickPdfThumbnailProvider*, *sh::tools::thumbnailproviders::PlaintextThumbnailPro*

## Public Functions

void **getThumbnail** (*sh::filesystem::Operation \*operation*, std::shared\_ptr<*sh::filesystem::FileSystemNode*>  
*node*, QString *contentType*, int *width*, int *height*, QIcon *\*icon*) = 0

**class UserDirLock**

*#include <userdirlock.h>* Locks the Shallot user directory during exclusive usage.

This lock is rather heavy-weight and should be used only if necessary!

## Public Functions

**UserDirLock** ()

Is intended to be directly constructed from everywhere.

**~UserDirLock** ()

## Private Members

bool **dounlock**

QString **slockpref**

QString **slockfile**

## Private Static Attributes

QThread **\*currentThread** = 0

QMutex **mutex**

**class VisibleViews** : public *sh::base::Singleton*

*#include <visibleviews.h>* Used for keeping track of which fileviews show which directories.

Whenever the view goes to other directories, this manager gets notified and redirects the notification to some other parts of the program.



### Public Functions

**AbstractAccountsProvider** ( )

**~AbstractAccountsProvider** ( )

void **findAccounts** (std::shared\_ptr<Account> pattern, QList<std::shared\_ptr<Account>> &result) = 0  
Finds account info by a given pattern.

bool **storeAccount** (std::shared\_ptr<Account> account) = 0  
Stores account infos.

**class Account**

*#include <accountsmanager.h> Account* data (for accessing network drives).

### Public Functions

**Account** ( ) = default

**Account** (const Account &other) = default

### Public Members

QString **username**

QString **domain**

QString **server**

int **serverport** = -1

QString **path**

QString **protocol**

QString **authtype**

QByteArray **authinfo**

**class AccountsManager** : public sh::base::Singleton

*#include <accountsmanager.h>* Storage for account data (for accessing network drives).

### Public Functions

QList<std::shared\_ptr<Account>> **findAccounts** (std::shared\_ptr<Account> pattern)  
Finds account infos by a given pattern.

void **storeAccount** (std::shared\_ptr<Account> account)  
Stores account infos.

void **registerProvider** (std::shared\_ptr<AbstractAccountsProvider> provider)  
Registers a new accounts provider.

void **doInitialize** ()  
Executes singleton initialization.

void **doShutdown** ()  
Executes singleton shutdown.



```
void shutdown ()
    Shutdown down this singleton.

bool isAlive ()
    Returns if this singleton is alive (true until its shutdown begins).
```

### Private Functions

```
AccountsManager ()
```

### Private Members

```
QList<std::shared_ptr<AbstractAccountsProvider>> accountsproviders

QMutex mutex
```

```
class FallbackAccountsProvider : public sh::tools::accounts::AbstractAccountsProvider
    #include <fallbackaccountsprovider.h> A fallback accounts provider which at least stores user names
    (i.e. no passwords) locally on disk.
```

### Public Functions

```
FallbackAccountsProvider ()

void findAccounts (std::shared_ptr<Account> pattern, QList<std::shared_ptr<Account>>
                    &result)
    Finds account info by a given pattern.

bool storeAccount (std::shared_ptr<Account> account)
    Stores account infos.
```

### Public Static Functions

```
void doInitialize ()

void doShutdown ()
```

### Private Functions

```
QList<std::shared_ptr<Account>> _find_rem_account_helper (std::shared_ptr<Account>
                                                         pattern, bool remove)
```

### Private Members

```
QString accountsdir
```

```
class LibsecretAccountsProvider : public sh::tools::accounts::AbstractAccountsProvider
    #include <libsecretaccountsprovider.h> An accounts provider based on gnome-keyring.
```



```
QList<std::shared_ptr<OpenMethod>> getOpenMethods (QString mimetype,
                                                    QList<std::shared_ptr<sh::filesystem::FilesystemNode>>
                                                    nodes)
```

Determines how to open a file with a given mimetype with an external program.

```
QList<std::shared_ptr<OpenMethod>> getAllOpenMethods ()
```

Returns a list of all known infos how to open a file external programs.

This has roughly one entry for each installed program on the user's system, which can graphically open files.

```
void doInitialize ()
```

Executes singleton initialization.

```
void doShutdown ()
```

Executes singleton shutdown.

```
void shutdown ()
```

Shutdown down this singleton.

```
bool isAlive ()
```

Returns if this singleton is alive (true until its shutdown begins).

## Private Functions

```
FileTypeManager (QObject *parent = 0)
```

## Private Members

```
QList<std::shared_ptr<MimetypeDeterminationStrategy>> _mimetypeDeterminationMethods
```

```
QList<std::shared_ptr<OpenMethodDeterminationStrategy>> _openMethodDeterminationMethods
```

```
QList<std::shared_ptr<MimetypeInformationRetrievalStrategy>> _mimetypeInformationRetrievalMethods
```

```
QMutex _mutex
```

```
class MimetypeDeterminationStrategy
```

*#include <filetypemanager.h>* Abstract base class for a mimetype determination strategy.

Subclassed by *sh::tools::filetypes::FreedesktopOrgToolsMimetypeDeterminationStrategy*,

*sh::tools::filetypes::SuffixListMimetypeDeterminationStrategy*, *sh::tools::filetypes::UnixFileToolMimetypeDeterminationStrategy*

## Public Functions

```
QHash<std::shared_ptr<const sh::filesystem::Eurl>, QString> determineMimetype (sh::filesystem::Operation
                                                                              *op,
                                                                              QList<std::shared_ptr<
                                                                              sh::filesystem::Eurl>>
                                                                              items)
=
0
```

```
~MimetypeDeterminationStrategy ()
```

```
class MimetypeInformationRetrievalStrategy
```

*#include <filetypemanager.h>* Abstract base class for mimetype information retrieval strategy.

Subclassed by *sh::tools::filetypes::FreeDesktopOrgMimetypeInformationRetrievalStrategy*

### Public Functions

QStringList **getMimeTypeSubclasses** (QString *mimetype*)

**~MimeTypeInformationRetrievalStrategy** ()

**class OpenMethodDeterminationStrategy**

*#include <filetypemanager.h>* Abstract base class for a ‘open method’ determination strategy.

Subclassed by *sh::tools::filetypes::FreedesktopOrgToolsOpenMethodDeterminationStrategy*,  
*sh::tools::filetypes::UserDefinedOpenMethodDeterminationStrategy*

### Public Functions

QList<std::shared\_ptr<*OpenMethod*>> **getOpenMethods** (QString *mimetype*,  
QList<std::shared\_ptr<*sh::filesystem::FilesystemNodes*>> *nodes*) = 0

**~OpenMethodDeterminationStrategy** ()

**class FreeDesktopOrgMimeTypeInformationRetrievalStrategy : public *sh::tools::filetypes::FileType***

*#include <freedesktoporgmimetypeinformationretrievalstrategy.h>* Tries to determine some mime-type information with the freedesktop.org specs.

### Public Functions

**FreeDesktopOrgMimeTypeInformationRetrievalStrategy** ()

QStringList **getMimeTypeSubclasses** (QString *mimetype*)

### Private Members

QHash<QString, QStringList> **\_mimeSubclassOf**

QMutex **\_mutex**

QWaitCondition **\_cond\_initied**

bool **\_initied** = false

**class MimeTypeInfo : public QXmlDefaultHandler**

### Public Functions

**MimeTypeInfo** () = default

bool **startElement** (const QString &*namespaceURI*, const QString &*localName*,  
const QString &*qName*, const QXmlAttributes &*atts*)

bool **endElement** (const QString &*namespaceURI*, const QString &*localName*, const  
QString &*qName*)

## Public Members

QStringList **subClassOf**

**class FreedesktopOrgToolsMimetypeDeterminationStrategy** : public *sh::tools::filetypes::FileTypeManager*  
*#include <freedesktoporgtoolsmimetypedeterminationmethod.h>* Tries to determine a file's mimetype  
 with the freedesktop.org tools.

## Public Functions

**FreedesktopOrgToolsMimetypeDeterminationStrategy** (*sh::tools::filetypes::FileTypeManager*  
*\*manager*)

QHash<std::shared\_ptr<const *sh::filesystem::Eurl*>, QString> **determineMimetype** (*sh::filesystem::Operation*  
*\*op*,  
 QList<std::shared\_ptr<const *sh::filesystem::Eurl*>>  
*items*)

## Private Members

QMutex **\_mutex**

QString **\_pathToFileTool**

const QRegularExpression **\_reMimetype**

## Private Static Attributes

std::shared\_ptr<*sh::configuration::ConfigurationValue*> **cfgvalXdgmimePath** = *sh::configuration::ConfigurationManager*

**class FreedesktopOrgToolsOpenMethodDeterminationStrategy** : public *sh::tools::filetypes::FileTypeManager*  
*#include <freedesktoporgtoolsopenmethoddeterminationmethod.h>* Tries to determine a 'open  
 method' for a file with the freedesktop.org tools.

## Public Functions

**FreedesktopOrgToolsOpenMethodDeterminationStrategy** ()

**~FreedesktopOrgToolsOpenMethodDeterminationStrategy** ()

QList<std::shared\_ptr<*sh::tools::filetypes::OpenMethod*>> **getOpenMethods** (QString  
*mimetype*,  
 QList<std::shared\_ptr<*sh::filesystem::Nodes*>>  
*nodes*)

### Private Functions

```
QString _parseValue (QStringList content, QString key)  
void _parseExecLine (QString execline, QString *command, QStringList *arguments)  
QString _escapeExecLineToken (QString s)
```

### Private Members

```
QMultiMap<QString, ApplicationEntry*> _mimetype2applicationEntry  
QMutex _mutex  
QWaitCondition _cond_initied  
bool _initied = false  
struct ApplicationEntry
```

### Public Members

```
QString name  
QStringList mimetypes  
QString command  
QStringList commandargs  
QIcon icon  
bool hidden
```

```
struct OpenMethod
```

```
#include <filetypemanager.h> Commandline and infos for opening a file with an external program.
```

### Public Functions

```
OpenMethod (QString name, QString command, QStringList arguments, QIcon icon = QIcon(),  
             int precedence = 0)
```

### Public Members

```
const QString name  
const QString command  
const QStringList arguments  
const QIcon icon  
const int precedence
```

```
class SuffixListMimetypeDeterminationStrategy : public sh::tools::filetypes::FileTypeManager::Mime  
#include <suffixlistmimetypedeterminationmethod.h> Tries to determine a file's mimetype with an  
internal lookup table of file extensions.
```

## Public Functions

**SuffixListMimetypeDeterminationStrategy** ()

QHash<std::shared\_ptr<const *sh::filesystem::Eurl*>, QString> **determineMimetype** (*sh::filesystem::Operation* \*op, QList<std::shared\_ptr<const *sh::filesystem::Eurl*>> items)

## Private Members

QHash<QString, QString> **\_suffixToMimetype**

QMutex **\_mutex**

**class UnixFileToolMimetypeDeterminationStrategy : public *sh::tools::filetypes::FileTypeManager::MimetypeDeterminationStrategy***  
*#include <unixfiletoolmimetyperedeterminationmethod.h>* Tries to determine a file's mimetype with the unix file tool.

## Public Functions

**UnixFileToolMimetypeDeterminationStrategy** (*sh::tools::filetypes::FileTypeManager* \*manager)

QHash<std::shared\_ptr<const *sh::filesystem::Eurl*>, QString> **determineMimetype** (*sh::filesystem::Operation* \*op, QList<std::shared\_ptr<const *sh::filesystem::Eurl*>> items)

## Public Static Attributes

std::shared\_ptr<*sh::configuration::ConfigurationValue*> **cfgvalFilePath** = *sh::configuration::ConfigurationManager::filePath*

## Private Members

QString **\_pathToFileTool**

const QRegularExpression **\_reMimetype**

QMutex **\_mutex**

**class UserDefinedOpenMethodDeterminationStrategy : public *sh::tools::filetypes::FileTypeManager::MimetypeDeterminationStrategy***  
*#include <userdefinedopenmethoddeterminationstrategy.h>* Tries to determine a 'open method' for a file by information the user stored before.

## Public Functions

```
QList<std::shared_ptr<sh::tools::filetypes::OpenMethod>> getOpenMethods (QString
                                                                    mimetype,
                                                                    QList<std::shared_ptr<sh::filesystem::
                                                                    nodes>
                                                                    override

void storeCustomOpenMethod (QList<std::shared_ptr<sh::filesystem::FileSystemNode>>
                                                                    nodes,
                                                                    QString
                                                                    mimetype,
                                                                    std::shared_ptr<sh::tools::filetypes::OpenMethod>
                                                                    m,
                                                                    bool
                                                                    rememberForMimetype,
                                                                    std::shared_ptr<const
                                                                    sh::filesystem::Eurl>
                                                                    rememberForDirectory,
                                                                    bool
                                                                    rememberForFile)

    Stores a custom open method.

void doInitialize ()
    Executes singleton initialization.

void doShutdown ()
    Executes singleton shutdown.

void shutdown ()
    Shutdown down this singleton.

bool isAlive ()
    Returns if this singleton is alive (true until its shutdown begins).
```

## Private Functions

```
UserDefinedOpenMethodDeterminationStrategy ()
```

## Private Members

```
QMutex _mutex
```

```
namespace thumbnailproviders
```

Thumbnail providers.

Subclasses of *sh::tools::ThumbnailProvider* (and possibly some auxiliary stuff). Those classes are used for generating thumbnail pictures for files.

```
class DefaultImageThumbnailProvider : public sh::tools::ThumbnailProvider
    #include <defaultimagethumbnailprovider.h> Thumbnail provider for images.
```

## Public Functions

```
DefaultImageThumbnailProvider () = default
```

```
void getThumbnail (sh::filesystem::Operation *operation, std::shared_ptr<sh::filesystem::FileSystemNode>
                                                                    node,
                                                                    QString
                                                                    contentType,
                                                                    int
                                                                    width,
                                                                    int
                                                                    height,
                                                                    QIcon
                                                                    *icon)
                                                                    override
```



### Public Static Functions

void **doInitialize** ()

void **doShutdown** ()

**class FfmpegVideoThumbnailProvider** : public QObject, public *sh::tools::ThumbnailProvider*  
*#include <ffmpegvideothumbnailprovider.h>* Thumbnail provider for videos using the ffmpeg tool.

### Public Functions

**FfmpegVideoThumbnailProvider** ()

void **getThumbnail** (*sh::filesystem::Operation* \*operation, std::shared\_ptr<*sh::filesystem::FilesystemNode*>  
node, QString contentType, int width, int height, QIcon \*icon)  
**override**

### Public Static Functions

void **doInitialize** ()

void **doShutdown** ()

### Private Members

QString **pathToFfmpegTool**

QString **pathToFfprobeTool**

**const** QRegularExpression **reDuration**

QMutex **mutexReDuration**

### Private Static Attributes

std::shared\_ptr<*sh::configuration::ConfigurationValue*> **cfgvalFfmpegPath** = *sh::configuration::ConfigurationManager::instance().getConfigurationValue(FfmpegPath)*

**class ImageMagickPdfThumbnailProvider** : public QObject, public *sh::tools::ThumbnailProvider*  
*#include <imagemagickpdfthumbnailprovider.h>* Thumbnail provider for videos using the ffmpeg tool.

### Public Functions

**ImageMagickPdfThumbnailProvider** ()

void **getThumbnail** (*sh::filesystem::Operation* \*operation, std::shared\_ptr<*sh::filesystem::FilesystemNode*>  
node, QString contentType, int width, int height, QIcon \*icon)  
**override**

### Public Static Functions

void **doInitialize** ()

void **doShutdown** ()

### Private Members

QString **pathToImagemagickConvertTool**

### Private Static Attributes

std::shared\_ptr<sh::configuration::ConfigurationValue> **cfgvalImagemagickConvertPath** = sh::configuration::

**class PlaintextThumbnailProvider** : public sh::tools::ThumbnailProvider  
#include <plaintextthumbnailprovider.h> Thumbnail provider for plain text.

### Public Functions

**PlaintextThumbnailProvider** ()

void **getThumbnail** (sh::filesystem::Operation \*operation, std::shared\_ptr<sh::filesystem::FilesystemNode>  
node, QString contentType, int width, int height, QIcon \*icon)  
**override**

### Public Static Functions

void **doInitialize** ()

void **doShutdown** ()

### Private Members

QColor **brandingcolor**

## 10.2.22 Namespace sh::tools::accounts

**namespace sh::tools::accounts**

*Account* and password storage.

See *sh::tools::accounts::AccountsManager* for more.

**class AbstractAccountsProvider**

#include <abstractaccountsprovider.h> Abstract base class for accounts provider.

Implement this class (and register an instance of it) in order to add support for something like a password/account manager.

Subclassed by *sh::tools::accounts::FallbackAccountsProvider*, *sh::tools::accounts::LibsecretAccountsProvider*

## Public Functions

**AbstractAccountsProvider** ()

**~AbstractAccountsProvider** ()

void **findAccounts** (std::shared\_ptr<Account> pattern, QList<std::shared\_ptr<Account>> &result) = 0  
Finds account info by a given pattern.

bool **storeAccount** (std::shared\_ptr<Account> account) = 0  
Stores account infos.

**class Account**

*#include <accountsmanager.h>* Account data (for accessing network drives).

## Public Functions

**Account** () = default

**Account** (const Account &other) = default

## Public Members

QString **username**

QString **domain**

QString **server**

int **serverport** = -1

QString **path**

QString **protocol**

QString **authtype**

QByteArray **authinfo**

**class AccountsManager**: public sh::base::Singleton

*#include <accountsmanager.h>* Storage for account data (for accessing network drives).

## Public Functions

QList<std::shared\_ptr<Account>> **findAccounts** (std::shared\_ptr<Account> pattern)  
Finds account infos by a given pattern.

void **storeAccount** (std::shared\_ptr<Account> account)  
Stores account infos.

void **registerProvider** (std::shared\_ptr<AbstractAccountsProvider> provider)  
Registers a new accounts provider.

void **doInitialize** ()  
Executes singleton initialization.

void **doShutdown** ()  
Executes singleton shutdown.

```
void shutdown ()  
    Shutdown down this singleton.  
  
bool isAlive ()  
    Returns if this singleton is alive (true until its shutdown begins).
```

### Private Functions

```
AccountsManager ()
```

### Private Members

```
QList<std::shared_ptr<AbstractAccountsProvider>> accountsproviders  
  
QMutex mutex
```

```
class FallbackAccountsProvider : public sh::tools::accounts::AbstractAccountsProvider  
    #include <fallbackaccountsprovider.h> A fallback accounts provider which at least stores user names (i.e.  
    no passwords) locally on disk.
```

### Public Functions

```
FallbackAccountsProvider ()  
  
void findAccounts (std::shared_ptr<Account> pattern, QList<std::shared_ptr<Account>> &re-  
    sult)  
    Finds account info by a given pattern.  
  
bool storeAccount (std::shared_ptr<Account> account)  
    Stores account infos.
```

### Public Static Functions

```
void doInitialize ()  
  
void doShutdown ()
```

### Private Functions

```
QList<std::shared_ptr<Account>> _find_rem_account_helper (std::shared_ptr<Account>  
    pattern, bool remove)
```

### Private Members

```
QString accountsdir  
  
class LibsecretAccountsProvider : public sh::tools::accounts::AbstractAccountsProvider  
    #include <libsecretaccountsprovider.h> An accounts provider based on gnome-keyring.
```

## Public Functions

**LibsecretAccountsProvider** ()

void **findAccounts** (std::shared\_ptr<Account> pattern, QList<std::shared\_ptr<Account>> &result)  
Finds account info by a given pattern.

bool **storeAccount** (std::shared\_ptr<Account> account)  
Stores account infos.

## Public Static Functions

void **doInitialize** ()

void **doShutdown** ()

## Private Members

QByteArray **sauthtype** = QString("authtype").toUtf8()

QByteArray **sdomain** = QString("domain").toUtf8()

QByteArray **spath** = QString("object").toUtf8()

QByteArray **sprotocol** = QString("protocol").toUtf8()

QByteArray **sserver** = QString("server").toUtf8()

QByteArray **sserverport** = QString("port").toUtf8()

QByteArray **susername** = QString("user").toUtf8()

## 10.2.23 Namespace sh::tools::filetypes

**namespace** *sh::tools::filetypes*

Tools for determining a file's type (pdf, image, mp3, et al) and ways how to deal with that.

See *sh::tools::filetypes::FileTypeManager* for more.

**class** **FileTypeManager** : public QObject, public *sh::base::Singleton*  
*#include <filetypemanager.h>* Utilities for dealing with file types.

It can determine the type of a file (png, plaintext, html, ...), it can provide information about how to open them with an external program, and more.

For most tasks it uses a pluggable interface. Actual implementations of strategies for those tasks reside in separate classes in this namespace.

## Public Functions

QString **determineMimetype** (*sh::filesystem::Operation* \*op, std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)

Determines the mimetype for one file.

QHash<std::shared\_ptr<const *sh::filesystem::Eurl*>, QString> **determineMimetype** (*sh::filesystem::Operation* \*op, QList<std::shared\_ptr<const *sh::filesystem::Eurl*>> items)

Determines the mimetypes for a list of files.

QList<std::shared\_ptr<*OpenMethod*>> **getOpenMethods** (QString mimetype, QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> nodes)

Determines how to open a file with a given mimetype with an external program.

QList<std::shared\_ptr<*OpenMethod*>> **getAllOpenMethods** ()

Returns a list of all known infos how to open a file external programs.

This has roughly one entry for each installed program on the user's system, which can graphically open files.

void **doInitialize** ()

Executes singleton initialization.

void **doShutdown** ()

Executes singleton shutdown.

void **shutdown** ()

Shutdown down this singleton.

bool **isAlive** ()

Returns if this singleton is alive (true until its shutdown begins).

## Private Functions

**FileTypeManager** (QObject \*parent = 0)

## Private Members

QList<std::shared\_ptr<*MimetypeDeterminationStrategy*>> **\_mimetypeDeterminationMethods**

QList<std::shared\_ptr<*OpenMethodDeterminationStrategy*>> **\_openMethodDeterminationMethods**

QList<std::shared\_ptr<*MimetypeInformationRetrievalStrategy*>> **\_mimetypeInformationRetrievalMethods**

QMutex **\_mutex**

**class MimetypeDeterminationStrategy**

#include <filetypemanager.h> Abstract base class for a mimetype determination strategy.

Subclassed by *sh::tools::filetypes::FreedesktopOrgToolsMimetypeDeterminationStrategy*,

*sh::tools::filetypes::SuffixListMimetypeDeterminationStrategy*, *sh::tools::filetypes::UnixFileToolMimetypeDeterminationStrategy*

## Public Functions

```
QHash<std::shared_ptr<const sh::filesystem::Eurl>, QString> determineMimetype (sh::filesystem::Operation
    *op,
    QList<std::shared_ptr<const sh::filesystem::Eurl>>
    items)
    = 0
```

```
~MimetypeDeterminationStrategy ()
```

```
class MimetypeInformationRetrievalStrategy
```

```
#include <filetypemanager.h> Abstract base class for mimetype information retrieval strategy.
```

```
Subclassed by sh::tools::filetypes::FreeDesktopOrgMimetypeInformationRetrievalStrategy
```

## Public Functions

```
QStringList getMimetypeSubclasses (QString mimetype)
```

```
~MimetypeInformationRetrievalStrategy ()
```

```
class OpenMethodDeterminationStrategy
```

```
#include <filetypemanager.h> Abstract base class for a ‘open method’ determination strategy.
```

```
Subclassed by sh::tools::filetypes::FreedesktopOrgToolsOpenMethodDeterminationStrategy,
sh::tools::filetypes::UserDefinedOpenMethodDeterminationStrategy
```

## Public Functions

```
QList<std::shared_ptr<OpenMethod>> getOpenMethods (QString mimetype,
    QList<std::shared_ptr<sh::filesystem::FilesystemNode>>
    nodes) = 0
```

```
~OpenMethodDeterminationStrategy ()
```

```
class FreeDesktopOrgMimetypeInformationRetrievalStrategy : public sh::tools::filetypes::FileTypeManager
```

```
#include <freedesktoporgmimetypeinformationretrievalstrategy.h> Tries to determine some mimetype in-
formation with the freedesktop.org specs.
```

## Public Functions

```
FreeDesktopOrgMimetypeInformationRetrievalStrategy ()
```

```
QStringList getMimetypeSubclasses (QString mimetype)
```

## Private Members

```
QHash<QString, QStringList> _mimeSubclassOf
```

```
QMutex _mutex
```

```
QWaitCondition _cond_initd
```

```
bool _initd = false
```

```
class MimetypeInfo : public QXmlDefaultHandler
```

## Public Functions

**MimetypeInfo** () = default

bool **startElement** (const QString &namespaceURI, const QString &localName, const QString &qName, const QDomAttributes &atts)

bool **endElement** (const QString &namespaceURI, const QString &localName, const QString &qName)

## Public Members

QStringList **subClassOf**

**class FreedesktopOrgToolsMimetypeDeterminationStrategy** : public *sh::tools::filetypes::FileTypeManager*  
#include <freedesktoporgtoolsmimetyperedeterminationmethod.h> Tries to determine a file's mimetype with the freedesktop.org tools.

## Public Functions

**FreedesktopOrgToolsMimetypeDeterminationStrategy** (*sh::tools::filetypes::FileTypeManager* \*manager)

QHash<std::shared\_ptr<const *sh::filesystem::Eurl*>, QString> **determineMimetype** (*sh::filesystem::Operation* \*op, QList<std::shared\_ptr<const *sh::filesystem::Eurl*>> items)

## Private Members

QMutex **\_mutex**

QString **\_pathToFileTool**

const QRegularExpression **\_reMimetype**

## Private Static Attributes

std::shared\_ptr<*sh::configuration::ConfigurationValue*> **cfgvalXdgmimePath** = *sh::configuration::ConfigurationManager*

**class FreedesktopOrgToolsOpenMethodDeterminationStrategy** : public *sh::tools::filetypes::FileTypeManager*  
#include <freedesktoporgtoolsopenmethoddeterminationmethod.h> Tries to determine a 'open method' for a file with the freedesktop.org tools.



## Public Functions

**FreedesktopOrgToolsOpenMethodDeterminationStrategy** ()

**~FreedesktopOrgToolsOpenMethodDeterminationStrategy** ()

**QList<std::shared\_ptr<sh::tools::filetypes::OpenMethod>> getOpenMethods** (QString *mimetype*,  
QList<std::shared\_ptr<sh::filesystem::File  
nodes)

## Private Functions

QString **\_parseValue** (QStringList *content*, QString *key*)

void **\_parseExecLine** (QString *execline*, QString \**command*, QStringList \**arguments*)

QString **\_escapeExecLineToken** (QString *s*)

## Private Members

QMultiMap<QString, *ApplicationEntry*\*> **\_mimetype2applicationEntry**

QMutex **\_mutex**

QWaitCondition **\_cond\_initied**

bool **\_initied** = false

**struct ApplicationEntry**

### Public Members

QString **name**

QStringList **mimetypes**

QString **command**

QStringList **commandargs**

QIcon **icon**

bool **hidden**

**struct OpenMethod**

*#include <filetypemanager.h>* Commandline and infos for opening a file with an external program.

## Public Functions

**OpenMethod** (QString *name*, QString *command*, QStringList *arguments*, QIcon *icon* = QIcon(), int  
*precedence* = 0)

### Public Members

```
const QString name
const QString command
const QStringList arguments
const QIcon icon
const int precedence
```

```
class SuffixListMimetypeDeterminationStrategy : public sh::tools::filetypes::FileTypeManager::MimetypeDeterminationStrategy
#include <suffixlistmimetyperedeterminationmethod.h> Tries to determine a file's mimetype with an internal
lookup table of file extensions.
```

### Public Functions

```
SuffixListMimetypeDeterminationStrategy ()

QHash<std::shared_ptr<const sh::filesystem::Eurl>, QString> determineMimetype (sh::filesystem::Operation
                                                                    *op,
                                                                    QList<std::shared_ptr<const
                                                                    sh::filesystem::Eurl>>
                                                                    items)
```

### Private Members

```
QHash<QString, QString> _suffixToMimetype
QMutex _mutex
```

```
class UnixFileToolMimetypeDeterminationStrategy : public sh::tools::filetypes::FileTypeManager::MimetypeDeterminationStrategy
#include <unixfiletoolmimetyperedeterminationmethod.h> Tries to determine a file's mimetype with the
unix file tool.
```

### Public Functions

```
UnixFileToolMimetypeDeterminationStrategy (sh::tools::filetypes::FileTypeManager
                                                                    *manager)

QHash<std::shared_ptr<const sh::filesystem::Eurl>, QString> determineMimetype (sh::filesystem::Operation
                                                                    *op,
                                                                    QList<std::shared_ptr<const
                                                                    sh::filesystem::Eurl>>
                                                                    items)
```

## Public Static Attributes

`std::shared_ptr<sh::configuration::ConfigurationValue> cfgvalFilePath = sh::configuration::ConfigurationManager::i`

## Private Members

`QString _pathToFileTool`

`const QRegularExpression _reMimetype`

`QMutex _mutex`

**class UserDefinedOpenMethodDeterminationStrategy** : public *sh::tools::filetypes::FileTypeManager::OpenM*  
*#include <userdefinedopenmethoddeterminationstrategy.h>* Tries to determine a 'open method' for a file  
 by information the user stored before.

## Public Functions

`QList<std::shared_ptr<sh::tools::filetypes::OpenMethod>> getOpenMethods (QString mimetype,`  
`QList<std::shared_ptr<sh::filesystem::File`  
`nodes)`  
**override**

`void storeCustomOpenMethod (QList<std::shared_ptr<sh::filesystem::FilesystemNode>>`  
`nodes, QString mimetype, std::shared_ptr<sh::tools::filetypes::OpenMethod>`  
`m, bool rememberForMimetype, std::shared_ptr<const`  
`sh::filesystem::Eurl> rememberForDirectory, bool remember-`  
`ForFile)`

Stores a custom open method.

`void doInitialize ()`  
 Executes singleton initialization.

`void doShutdown ()`  
 Executes singleton shutdown.

`void shutdown ()`  
 Shutdown down this singleton.

`bool isAlive ()`  
 Returns if this singleton is alive (true until its shutdown begins).

## Private Functions

`UserDefinedOpenMethodDeterminationStrategy ()`

## Private Members

QMutex `_mutex`

## 10.2.24 Namespace `sh::tools::thumbnailproviders`

**namespace** `sh::tools::thumbnailproviders`

Thumbnail providers.

Subclasses of `sh::tools::ThumbnailProvider` (and possibly some auxiliary stuff). Those classes are used for generating thumbnail pictures for files.

**class** `DefaultImageThumbnailProvider` : **public** `sh::tools::ThumbnailProvider`  
*#include <defaultimagethumbnailprovider.h>* Thumbnail provider for images.

### Public Functions

`DefaultImageThumbnailProvider` () = default

void `getThumbnail` (`sh::filesystem::Operation` \**operation*, std::shared\_ptr<`sh::filesystem::FileSystemNode`> *node*, QString *contentType*, int *width*, int *height*, QIcon \**icon*) **override**

### Public Static Functions

void `doInitialize` ()

void `doShutdown` ()

**class** `FfmpegVideoThumbnailProvider` : **public** QObject, **public** `sh::tools::ThumbnailProvider`  
*#include <ffmpegvideothumbnailprovider.h>* Thumbnail provider for videos using the ffmpeg tool.

### Public Functions

`FfmpegVideoThumbnailProvider` ()

void `getThumbnail` (`sh::filesystem::Operation` \**operation*, std::shared\_ptr<`sh::filesystem::FileSystemNode`> *node*, QString *contentType*, int *width*, int *height*, QIcon \**icon*) **override**

### Public Static Functions

void `doInitialize` ()

void `doShutdown` ()

### Private Members

```
QString pathToFfmpegTool
QString pathToFfprobeTool
const QRegularExpression reDuration
QMutex mutexReDuration
```

### Private Static Attributes

```
std::shared_ptr<sh::configuration::ConfigurationValue> cfgvalFfmpegPath = sh::configuration::ConfigurationManager::getInstance().getFfmpegPath();

class ImageMagickPdfThumbnailProvider : public QObject, public sh::tools::ThumbnailProvider
#include <imagemagickpdfthumbnailprovider.h> Thumbnail provider for videos using the ffmpeg tool.
```

### Public Functions

```
ImageMagickPdfThumbnailProvider ()

void getThumbnail (sh::filesystem::Operation *operation, std::shared_ptr<sh::filesystem::FilesystemNode>
node, QString contentType, int width, int height, QIcon *icon) override
```

### Public Static Functions

```
void doInitialize ()
void doShutdown ()
```

### Private Members

```
QString pathToImagemagickConvertTool
```

### Private Static Attributes

```
std::shared_ptr<sh::configuration::ConfigurationValue> cfgvalImagemagickConvertPath = sh::configuration::ConfigurationManager::getInstance().getImagemagickConvertPath();

class PlaintextThumbnailProvider : public sh::tools::ThumbnailProvider
#include <plaintextthumbnailprovider.h> Thumbnail provider for plain text.
```

### Public Functions

```
PlaintextThumbnailProvider ()

void getThumbnail (sh::filesystem::Operation *operation, std::shared_ptr<sh::filesystem::FilesystemNode>
node, QString contentType, int width, int height, QIcon *icon) override
```

### Public Static Functions

void **doInitialize** ()

void **doShutdown** ()

### Private Members

QColor **brandingcolor**

## 10.2.25 Namespace sh::ui

**namespace** *sh::ui*

User interface.

This is the presentation layer. The classes directly from here are abstract; often even technically, but clearly from conceptual perspective. Find non-abstract user interface implementations in sub-namespaces.

### Typedefs

**typedef** QList<*sh::ui::FilePropertyDialogTabActionsView*\*> **TabViewStructWidgets**

**typedef** QPair<std::shared\_ptr<*sh::ui::FilePropertyDialogTab*>, *TabViewStructWidgets*> **TabViewStruct**

### Enums

**enum** **ExceptionDialogResult**

Enumeration of decisions the user can make in an *ExceptionDialog*.

*Values:*

**enumerator** **Shutdown**

**enumerator** **Close**

**enumerator** **Cancel**

**enumerator** **Retry**

**enum** **FileViewMode**

*Values:*

**enumerator** **IconView**

**enumerator** **ListView**

**class** **\_ActionExecutionInfoPanel\_HelperQObject** : **public** QObject

*#include* <actionexecutioninfopanel.h> A helper for some signal/slot stuff of in *ActionExecutionInfoPanel*.

## Public Functions

```
void _emit_clicked ()
void _emit_visibilityChanged ()
```

## Signals

```
void clicked ()
void visibilityChanged ()
```

```
class _FilePropertyDialogTabActionsView_HelperQObject : public QObject
#include <filepropertydialogtabactionsview.h> A helper for some signal/slot stuff of in FilePropertyDialogTabActionsView.
```

## Public Functions

```
void _emit_buttonTriggered (int i)
```

## Signals

```
void buttonTriggered (int i)
```

```
class _MainWindow_HelperQObject : public QObject
#include <mainwindow.h> A helper for some signal/slot stuff of in MainWindow.
```

## Public Functions

```
void _emit_currentDirectoryChanged ()
void _emit_globalViewOptionsChanged ()
void _emit_currentProfileChanged ()
void _emit_currentFileViewChanged ()
void _emit_fileViewCountChanged ()
void _emit_fileViewOptionsChanged (int i)
```

## Signals

```
void currentDirectoryChanged ()
void globalViewOptionsChanged ()
void currentProfileChanged ()
void currentFileViewChanged ()
void fileViewCountChanged ()
void fileViewOptionsChanged (int)
```

```
class _SearchPanelConfiguration_HelperQObject : public QObject
#include <searchpanelconfiguration.h> A helper for some signal/slot stuff of in SearchPanelConfiguration.
```

```
class AboutDialog : public sh::ui::Dialog
    #include <aboutdialog.h> The 'About' dialog.

    Subclassed by sh::ui::qt::QtAboutDialog, sh::ui::web::WebAboutDialog
```

## Public Functions

**AboutDialog()**

qint64 **dialogId()**

Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

bool **isInitiated()**

Returns if this dialog is initialized.

Must be called in main thread.

bool **wasAccepted()**

Returns if this dialog was 'accepted' by the user (typically by clicking some 'OK' button).

Must be called in main thread.

void **waitClosed()**

Wait until the user closed the dialog in some way.

May be called in any thread.

void **close()**

Closes the dialog.

Must be called in main thread.

bool **wasClosed()**

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \***manager()**

Returns the *DialogManager* which hosts this dialog.

```
class ActionExecutionInfoDialog : public sh::actions::ActionExecutionUserFeedback
    #include <actionexecutioninfodialog.h> Progress dialog for action executions.
```

Subclassed by *sh::ui::qt::QtActionExecutionInfoDialog*, *sh::ui::web::WebActionExecutionInfoDialog*

## Public Types

enum **MessageBoxButton**

Buttons in a message box from *ActionExecutionUserFeedback*.

Values:

enumerator **NONE** = 0

enumerator **OK** = 1 << 0

enumerator **Continue** = 1 << 1

enumerator **Cancel** = 1 << 2



```

enumerator Retry = 1 << 3
enumerator Yes = 1 << 4
enumerator No = 1 << 5

```

## Public Functions

**ActionExecutionInfoDialog** (*sh::actions::ActionExecutionInfo* \*info)

void **setDetails** (QString fv, QString fob, QString tv, QString tob) = 0  
Sets the item details text ('from a/foo.jpg', 'to b/foo.jpg'). .

void **setHead** (QString txt) = 0  
Sets the header text. .

void **setProgress** (bool isDeterminate, quint64 done, quint64 all, QString text) = 0  
Sets the progress. .

bool **isLogicallyVisible** ()  
Returns if this dialog is logically visible (i.e. set visible by the action).

void **setLogicallyVisible** (bool v)  
Sets if this dialog is logically visible (i.e. set visible by the action). .

bool **isBackground** ()  
Returns if this dialog is currently in background mode (i.e. not visible).

void **setBackground** (bool v)  
Sets if this dialog is currently in background mode (i.e. not visible). .

void **setForceForeground** (bool v)  
Sets if this dialog is currently forced to be visible in foreground. .

int **messageBox** (QString text, QList<QString> answers, QString icon = QString(), int defaultanswer = -1, int cancelanswer = -1, QList<QString> answericons = QList<QString>()) = 0

int **inputBox** (QString text, QList<QString> answers, QString \*value, QString icon = QString(), int defaultanswer = -1, int cancelanswer = -1, int valuePreselectFrom = -1, int valuePreselectTo = -1) = 0

int **multilineInputBox** (QString text, QList<QString> answers, QString \*value, QString icon = QString(), int defaultanswer = -1, int cancelanswer = -1) = 0

int **simpleChooserGridform** (QString text, GridformEntries \*entries, QStringList answers, int defaultanswer = -1, int cancelanswer = -1) = 0

bool **credentialsDialog** (QString text, bool showDomain, bool showUsername, bool showPassword, bool showAnonymous, bool showRemember, QString \*domain, QString \*username, QString \*password, bool \*isAnonymous, bool \*isRemember) = 0

bool **unixPermissionsDialog** (bool \*userMayRead, bool \*userMayWrite, bool \*userMayExecute, bool \*groupMayRead, bool \*groupMayWrite, bool \*groupMayExecute, bool \*othersMayRead, bool \*othersMayWrite, bool \*othersMayExecute, bool \*sticky, bool \*setuid, bool \*setgid, QStringList users, QStringList groups, QString \*ownerUser, QString \*ownerGroup) = 0

QString **simpleInputBox** (QString text, QString deflt, int valuePreselectFrom = -1, int valuePreselectTo = -1)

```
int simpleMessageBox (QString text, int buttons = (int)MessageBoxButton::OK, QString icon =  
    QString(), int defaultbutton = (MessageBoxButton)0, int cancelbutton =  
    (MessageBoxButton)0)
```

```
class ActionExecutionInfoPanel
```

```
#include <actionexecutioninfopanel.h> Abstract class for a status bar info-panel for an action execution.
```

```
Subclassed by sh::ui::qt::QtActionExecutionInfoPanel, sh::ui::web::WebActionExecutionInfoPanel
```

### Public Functions

```
void setLabel (QString s) = 0
```

```
    Sets the label text. .
```

```
void setProgress (bool isProgressDeterminate, quint64 progressDone, quint64 progressAll) = 0
```

```
    Sets the progress. .
```

```
void setForceForeground (bool v) = 0
```

```
    Sets if the associated action is currently forced to be visible in foreground. .
```

```
void setPanelVisible (bool v) = 0
```

```
    Sets if the panel is visible. .
```

```
void setWidth (int width) = 0
```

```
    Sets the panel with (in pixels). .
```

```
void onClicked (std::function<void>)
```

```
    > fcnQObject *owner = 0 Sets a handler for a click on the button (optionally bound to an owner life-  
    time).
```

```
void onDestroyed (std::function<void>)
```

```
    > fcnQObject *owner = 0 Sets a handler for panel removal (optionally bound to an owner lifetime).
```

```
void onVisibilityChanged (std::function<void>)
```

```
    > fcnQObject *owner = 0 Sets a handler for panel visibility changes (optionally bound to an owner  
    lifetime).
```

```
~ActionExecutionInfoPanel ()
```

```
class ColumnDimensions
```

```
#include <columndimensions.h> Auxiliary data structure for persistent column dimensions.
```

```
It stores the width of a column whenever it changes and restores it when needed (mostly when Shallot  
starts).
```

### Public Functions

```
ColumnDimensions (int index)
```

```
    Constructed only by the infrastructure and made available otherwise.
```

```
void store ()
```

```
bool remove ()
```

```
void setWidth (QString col, int width)
```

```
int getWidth (QString col, int defaultval)
```

## Public Members

int **index**

## Private Functions

std::unique\_ptr<QSettings> **db** ()

## Private Members

QHash<QString, int> **dims**

**class Dialog** : **public** std::enable\_shared\_from\_this<*Dialog*>  
*#include <dialog.h>* Abstract subclass for Shallot dialogs (in child windows).

Subclasses of *Dialog* represent particular dialog types (like *ManageBookmarksDialog*).

For each ui mode, there typically is an implementation for all of these types, implementing one of this dialog types subclasses (mentioned before) and also some ui mode specific class.

Note: Unless stated otherwise, all methods must be called from main thread!

Note: It's not allowed to show one web dialog instance more than once. After closing it, it's sole use is to fetch answer data from it.

Note: You should not create instances directly. It's required to use *DialogManager::createAndShowDialog()* for actually showing those dialogs. Even this should not be used directly, because call depends on the current ui mode. Use the show\*() methods in *sh::ui::MainWindow* for creating dialogs in a ui mode independent way.

Subclassed by *sh::ui::AboutDialog*, *sh::ui::ExceptionDialog*, *sh::ui::FilePropertyDialog*,  
*sh::ui::LogViewDialog*, *sh::ui::ManageBookmarksDialog*, *sh::ui::ManageProfilesDialog*,  
*sh::ui::OpenWithDialog*, *sh::ui::StoreProfileDialog*, *sh::ui::TuningDialog*

## Public Functions

**Dialog** ()

**~Dialog** ()

qint64 **dialogId** ()

Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

bool **isInited** ()

Returns if this dialog is initialized.

Must be called in main thread.

bool **wasAccepted** ()

Returns if this dialog was 'accepted' by the user (typically by clicking some 'OK' button).

Must be called in main thread.

void **waitClosed** ()  
Wait until the user closed the dialog in some way.  
May be called in any thread.

void **close** ()  
Closes the dialog.  
Must be called in main thread.

bool **wasClosed** ()  
Returns if this dialog was closed.  
Must be called in main thread.

*DialogManager* \***manager** ()  
Returns the *DialogManager* which hosts this dialog.

### Private Functions

void **setup** (*DialogManager* \**manager*, qint64 *dialogId*)  
Sets manager and id for this dialog (right after creation).

### Private Members

*DialogManager* \*\_**manager**  
qint64 **\_dialogId** = -1  
bool **\_closed** = false  
bool **\_initied** = false

### Friends

**friend class** MainWindow  
**friend class** DialogManager

### **class DialogManager**

*#include <dialog.h>* Creates and drives dialogs (in child windows), i.e. instances of *Dialog*.

Subclass it for implementing the low level mechanisms for handling dialogs in a particular ui mode (e.g. qt, web).

The infrastructure will use one of this implementations for creating and managing most kinds of dialogs.

Subclassed by *sh::ui::qt::QtDialogManager*, *sh::ui::web::WebDialogManager*

## Public Functions

`std::shared_ptr<Dialog> getDialogById (qint64 id)`

Returns a *Dialog* by dialog id.

Must be called in main thread.

`QList<std::shared_ptr<Dialog>> getAllDialogs ()`

Returns a list of all dialogs currently shown (in order of creation).

Must be called in main thread.

`QList<qint64> getAllDialogIds ()`

Returns a list of dialog ids of all dialogs currently shown (in order of creation).

Must be called in main thread.

`template<class TDlg, typename ...Args>`

`std::shared_ptr<TDlg> createAndShowDialog (Args... args)`

Creates and shows a *Dialog* by class and constructor parameters.

Those dialogs (TDlg) have to implement *Dialog* (typically a subclass of it, representing a particular dialog, like *FilePropertyDialog*), and also some ui mode (e.g. qt, web) specific class (depends on that ui mode).

You typically should not have to use it outside of *MainWindow* or subclasses. The *MainWindow* interface allows to create all available dialogs in a ui mode independent way.

May be called in any thread.

**Return** The created *Dialog* instance.

`~DialogManager ()`

## Private Functions

`void closeDialog (std::shared_ptr<Dialog> dialog)`

Closes this dialog, including internal bookkeeping.

Must be called in main thread.

`void initAndShowDialog (std::shared_ptr<Dialog> dialog)`

Initializes and shows a freshly created dialog.

Must be called in main thread.

`bool wasAccepted (std::shared_ptr<Dialog> dialog) = 0`

Returns if the dialog was ‘accepted’ by the user (typically by clicking some ‘OK’ button).

The result is undefined before the dialog was closed.

Must be called in main thread.

`void waitClosed (std::shared_ptr<Dialog> dialog) = 0`

Wait until the user closed the dialog in some way.

May be called in any thread.

`void stopDialog (std::shared_ptr<Dialog> dialog) = 0`

This method implements the ui mode specific mechanism for stopping (i.e. closing) a dialog.

Must be called in main thread.

```
void showDialog (std::shared_ptr<Dialog> dialog) = 0
```

This method implements the ui mode specific mechanism for showing a dialog.

Must be called in main thread.

### Private Members

```
QHash<qint64, std::shared_ptr<Dialog>> _openDialogs
```

```
qint64 _nextDialogId = 0
```

### Friends

```
friend class Dialog
```

```
class ExceptionDialog : public sh::ui::Dialog  
#include <exceptiondialog.h> The 'Exception' dialog.
```

Subclassed by *sh::ui::qt::QtExceptionDialog*, *sh::ui::web::WebExceptionDialog*

### Public Functions

```
ExceptionDialog (QString error1, QString error2, QString details, QString icon, bool  
mayRetry, bool mayClose, bool mayCancel, bool showLoglabelAndDetails)
```

#### Parameters

- *error1*: The 1st error text.
- *error2*: The 2nd error text.
- *details*: The error details.
- *icon*: An icon (as name resolveable by *sh::base::IconManager*).
- *mayRetry*: If it's allowed to retry.
- *mayClose*: If it's allowed to just close and continue without closing Shallot.
- *mayCancel*: If it's allowed to cancel, i.e. throwing a *sh::exceptions::CancelException*.
- *showLoglabelAndDetails*: If the dialog shall allow to read the details.

```
QString error1 ()
```

Returns the 1st error text.

```
QString error2 ()
```

Returns the 2nd error text.

```
QString details ()
```

Returns the error details.

```
QString icon ()
```

Returns the icon (as name resolveable by *sh::base::IconManager*).

```
bool mayRetry ()
```

Returns if it's allowed to retry.

```
bool mayClose ()
```

Returns if it's allowed to just close and continue without closing Shallot.

```
bool mayCancel ()
```

Returns if it's allowed to cancel, i.e. throwing a *sh::exceptions::CancelException*.

```
bool showLoglabelAndDetails ()
```

Returns if the dialog shall allow to read the details.

*ExceptionDialogResult* **answer** () = 0

Returns the answer the user has given in this dialog.

qint64 **dialogId** ()

Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

bool **isInitied** ()

Returns if this dialog is initialized.

Must be called in main thread.

bool **wasAccepted** ()

Returns if this dialog was ‘accepted’ by the user (typically by clicking some ‘OK’ button).

Must be called in main thread.

void **waitClosed** ()

Wait until the user closed the dialog in some way.

May be called in any thread.

void **close** ()

Closes the dialog.

Must be called in main thread.

bool **wasClosed** ()

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \***manager** ()

Returns the *DialogManager* which hosts this dialog.

## Private Members

QString **\_error1**

QString **\_error2**

QString **\_details**

QString **\_icon**

bool **\_mayRetry**

bool **\_mayClose**

bool **\_mayCancel**

bool **\_showLoglabelAndDetails**

**class FilePropertyDialog**: public *sh::ui::Dialog*

#include <filepropertydialog.h> The ‘File Properties’ dialog.

Subclassed by *sh::ui::qt::QtFilePropertyDialog*, *sh::ui::web::WebFilePropertyDialog*

## Public Functions

**~FilePropertyDialog** ()

**FilePropertyDialog** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> nodes)

void **refresh** () = 0

Refreshes the dialog content.

QList<std::shared\_ptr<*FilePropertyDialogTab*>> **tabs** ()

Returns a list of all tabs.

*sh::ui::FilePropertyDialogTabActionsView* \***widgetAt** (std::shared\_ptr<*FilePropertyDialogTab*>  
tab, int i)

Returns the widget from a tab at a certain index.

int **widgetCount** (std::shared\_ptr<*FilePropertyDialogTab*> tab)

Returns the number of widgets in a tab.

*sh::ui::FilePropertyDialogTabTableView* \***createTabViewTable** () = 0

Creates a new tab view table sub-widget.

*sh::ui::FilePropertyDialogTabTextView* \***createTabViewText** () = 0

Creates a new tab view text sub-widget.

*sh::ui::FilePropertyDialogTabIconTextBannerView* \***createTabViewIconTextBanner** () = 0

Creates a new tab view icon text banner sub-widget.

qint64 **dialogId** ()

Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

bool **isInitiated** ()

Returns if this dialog is initialized.

Must be called in main thread.

bool **wasAccepted** ()

Returns if this dialog was 'accepted' by the user (typically by clicking some 'OK' button).

Must be called in main thread.

void **waitClosed** ()

Wait until the user closed the dialog in some way.

May be called in any thread.

void **close** ()

Closes the dialog.

Must be called in main thread.

bool **wasClosed** ()

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \***manager** ()

Returns the *DialogManager* which hosts this dialog.



## Public Static Functions

void **addTabFactory** (int *i*, std::shared\_ptr<*FilePropertyDialogTabFactory*> *factory*)

Adds a *FilePropertyDialogTabFactory* so the Properties dialogs show its content.

See also the REGISTER\_FILEPROPERTYDIALOGTAB macro.

### Parameters

- *i*: An index which controls the display order. Use one of the REGISTER\_FILEPROPERTYDIALOGTAB\_INDEX\_\* values in *FilePropertyDialogTabFactory* as base values.

## Private Members

QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> **\_nodes**

QList<*TabViewStruct*> **\_tabs**

## Private Static Attributes

QHash<int, std::shared\_ptr<*FilePropertyDialogTabFactory*>> **\_propertytabs**

## Friends

**friend class** FilePropertyDialogTab

**class FilePropertyDialogTab** : public std::enable\_shared\_from\_this<*FilePropertyDialogTab*>  
*#include <filepropertydialog.h>* Abstract base class for one tab in the Properties dialog (containing a list of widgets).

Subclass and register this class for providing additional content in the Properties dialog.

Register it by using the REGISTER\_FILEPROPERTYDIALOGTAB macro (or by adding a *FilePropertyDialogTabFactory* via *FilePropertyDialog::addTabFactory*).

Subclassed by *sh::filepropertydialogtabs::FilePropertyDialogTabExtendedAttributes*,  
*sh::filepropertydialogtabs::FilePropertyDialogTabGeneral*, *sh::filepropertydialogtabs::FilePropertyDialogTabUnixPermissions*,  
*sh::filepropertydialogtabs::FilePropertyDialogTabWindows*, *sh::scripting::api::ApiFilePropertyDialogTab*

## Public Functions

QString **title** () = 0

The tab title. .

QString **titleWithoutMnemonic** ()

The tab title without mnemonic.

QList<QString> **properties** () = 0

Returns the list of property labels (one entry for each widget).

Each property is displayed with a header and a widget (created in createWidget) with some content (populated in updateWidget).

void **populateWidget** (int *i*, *FilePropertyDialogTabActionsView* \**widget*) = 0

Populates the tab widget for the *i*-th property.

Typically uses the FilePropertyDialog::create\* methods to create sub-widgets.

See also *dialog()*.

void **updateWidget** (int *i*, *FilePropertyDialogTabActionsView* \**widget*, *sh::filesystem::Operation* \**op*) = 0

Populates the widget for the *i*-th property with actual content. .

void **refresh** ()

Refreshes the complete content.

Typically called after some user actions in a property widget require that all the other content must be refreshed.

QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> **nodes** ()

The nodes to show.

*FilePropertyDialogTabActionsView* \***widgetAt** (int *i*)

Returns the widget for the *i*-th property.

int **widgetCount** ()

Returns the number of widgets.

std::shared\_ptr<*FilePropertyDialog*> **dialog** ()

Returns the associated dialog.

**~FilePropertyDialogTab** ()

## Private Members

std::weak\_ptr<*FilePropertyDialog*> **\_dialog**

## Friends

**friend class** FilePropertyDialog

**friend class** FilePropertyDialogTabFactoryByFunction

**friend class** sh::scripting::api::ApiFilePropertyDialogTabFactory

**class FilePropertyDialogTabActionsView**

*#include <filepropertydialogtabactionsview.h>* A tab view which shows a main widget in the main part and some buttons below.

Subclassed by *sh::ui::qt::QtFilePropertyDialogTabActionsView*, *sh::ui::web::WebFilePropertyDialog::WebFilePropertyDia*

## Public Functions

**FilePropertyDialogTabActionsView** ()

Is intended to be directly constructed from everywhere.

*FilePropertyDialogTabViewContent* \***content** ()

The main widget.

void **setContent** (*FilePropertyDialogTabViewContent* \**cnt*)

Sets the main widget. .

QList<QString> **buttons** ()

The list of buttons.

void **setButtons** (QList<QString> *buttons*)

Sets the list of buttons. .

void **setVisible** (bool v) = 0  
Sets the view visible or hidden. .

void **onButtonTriggered** (std::function<void>) int i  
> *fcn*, QObject \**owner* = 0 Sets a handler for a click on one of the buttons (optionally bound to an owner lifetime).

#### **class FilePropertyDialogTabFactory**

#include <filepropertydialog.h> Abstract factory for creating property dialog tabs for the selected nodes.

Register an instance of this class with *FilePropertyDialog::addTabFactory* in order to make a property tab implementation available. See also *FilePropertyDialogTabFactoryByFunction*.

Subclassed by *sh::scripting::api::ApiFilePropertyDialogTabFactory*, *sh::ui::FilePropertyDialogTabFactoryByFunction*

### Public Functions

std::shared\_ptr<*sh::ui::FilePropertyDialogTab*> **construct** (std::shared\_ptr<*sh::ui::FilePropertyDialog*>  
dialog) = 0  
Constructs a fresh instance of a *FilePropertyDialogTab* implementation. .

~**FilePropertyDialogTabFactory** ()

### Public Static Attributes

const int **REGISTER\_FILEPROPERTYDIALOGTAB\_INDEX\_CORE** = 10000

Base value for display indexes of core (i.e. maximum important) tabs.

Used in *FilePropertyDialog::addTabFactory*.

const int **REGISTER\_FILEPROPERTYDIALOGTAB\_INDEX\_VERYIMPORTANT** = 20000

Base value for display indexes of very important tabs.

Used in *FilePropertyDialog::addTabFactory*.

const int **REGISTER\_FILEPROPERTYDIALOGTAB\_INDEX\_IMPORTANT** = 30000

Base value for display indexes of important tabs.

Used in *FilePropertyDialog::addTabFactory*.

const int **REGISTER\_FILEPROPERTYDIALOGTAB\_INDEX\_NORMAL** = 40000

Base value for display indexes of tabs with medium importance.

Used in *FilePropertyDialog::addTabFactory*.

const int **REGISTER\_FILEPROPERTYDIALOGTAB\_INDEX\_EXOTIC** = 50000

Base value for display indexes of exotic (i.e. less important) tabs.

Used in *FilePropertyDialog::addTabFactory*.

#### **class FilePropertyDialogTabFactoryByFunction : public sh::ui::FilePropertyDialogTabFactory**

#include <filepropertydialog.h> Implementation of *FilePropertyDialogTabFactory* for making custom *FilePropertyDialogTab* implementations available.

This implementation relies on an external factory function which must be provided to the constructor.

Used internally inside the REGISTER\_FILEPROPERTYDIALOGTAB macro.

## Public Functions

**FilePropertyDialogTabFactoryByFunction** (std::function<std::shared\_ptr<sh::ui::FilePropertyDialogTab>> *fct*)

std::shared\_ptr<sh::ui::FilePropertyDialogTab> **construct** (std::shared\_ptr<sh::ui::FilePropertyDialog> *dialog*)

Constructs a fresh instance of a *FilePropertyDialogTab* implementation. .

## Public Static Attributes

**const int REGISTER\_FILEPROPERTYDIALOGTAB\_INDEX\_CORE** = 10000

Base value for display indexes of core (i.e. maximum important) tabs.

Used in *FilePropertyDialog::addTabFactory*.

**const int REGISTER\_FILEPROPERTYDIALOGTAB\_INDEX\_VERYIMPORTANT** = 20000

Base value for display indexes of very important tabs.

Used in *FilePropertyDialog::addTabFactory*.

**const int REGISTER\_FILEPROPERTYDIALOGTAB\_INDEX\_IMPORTANT** = 30000

Base value for display indexes of important tabs.

Used in *FilePropertyDialog::addTabFactory*.

**const int REGISTER\_FILEPROPERTYDIALOGTAB\_INDEX\_NORMAL** = 40000

Base value for display indexes of tabs with medium importance.

Used in *FilePropertyDialog::addTabFactory*.

**const int REGISTER\_FILEPROPERTYDIALOGTAB\_INDEX\_EXOTIC** = 50000

Base value for display indexes of exotic (i.e. less important) tabs.

Used in *FilePropertyDialog::addTabFactory*.

## Private Members

std::function< std::shared\_ptr< sh::ui::FilePropertyDialogTab >> **fct**

**class FilePropertyDialogTabIconTextBannerView** : public sh::ui::FilePropertyDialogTabViewContent  
#include <filepropertydialogtabactionsview.h> A tab view for showing texts and icons combined.

Subclassed by sh::ui::qt::QtFilePropertyDialogTabIconTextBannerView,  
sh::ui::web::WebFilePropertyDialog::WebFilePropertyDialogTabIconTextBannerView

## Public Functions

**FilePropertyDialogTabIconTextBannerView** ()

Is intended to be directly constructed from everywhere.

void **clear** () = 0

Clears the contents.

void **insertIcon** (QIcon *icon*, int *sizePt* = 24) = 0

Adds an icon.

void **insertText** (QString *text*) = 0

Adds a text.

**class FilePropertyDialogTabTableView** : public *sh::ui::FilePropertyDialogTabViewContent*  
*#include <filepropertydialogtabactionsview.h>* A tab view for showing key/value pairs.

Subclassed by *sh::ui::qt::QtFilePropertyDialogTabTableView*, *sh::ui::web::WebFilePropertyDialog::WebFilePropertyDialog*

## Public Types

**typedef** QPair<int, int> **ItemIndex**

## Public Functions

**FilePropertyDialogTabTableView** ()

Is intended to be directly constructed from everywhere.

void **setContent** (QList<QPair<QString, QString>> *content*) = 0

Sets the content.

QList<*ItemIndex*> **getSelectedItems** () = 0

Returns the selected cells.

QVariant **getItemValue** (int *r*, int *c*) = 0

Returns a value of a cell.

**class FilePropertyDialogTabTextView** : public *sh::ui::FilePropertyDialogTabViewContent*  
*#include <filepropertydialogtabactionsview.h>* A tab view for showing a text.

Subclassed by *sh::ui::qt::QtFilePropertyDialogTabTextView*, *sh::ui::web::WebFilePropertyDialog::WebFilePropertyDialog*

## Public Functions

**FilePropertyDialogTabTextView** ()

Is intended to be directly constructed from everywhere.

void **setContent** (QString *content*) = 0

Sets the content.

**class FilePropertyDialogTabViewContent**

*#include <filepropertydialogtabactionsview.h>* Abstract subclass for a tab view content.

Subclassed by *sh::ui::FilePropertyDialogTabIconTextBannerView*, *sh::ui::FilePropertyDialogTabTableView*,  
*sh::ui::FilePropertyDialogTabTextView*

## Public Functions

**FilePropertyDialogTabViewContent** ()

**~FilePropertyDialogTabViewContent** ()

**class FileView** : public QObject

*#include <fileview.h>* Abstract base class for a file view implementation (i.e. which shows the content of a directory somehow).

Subclassed by *sh::ui::qt::QtFileViewControl*, *sh::ui::web::WebFileView*

## Public Functions

**FileView()**

**~FileView()**

*sh::ui::ColumnDimensions* \***columnDimensions**() = 0

Returns the column dimensions handler. .

*sh::tools::HistoryTracker*<std::shared\_ptr<const *sh::filesystem::Eurl*>> \***historyTracker**() = 0

Returns the history tracker. .

*FileViewMode* **viewmode**() = 0

Returns the view mode (icons, list, ... ?). .

void **setViewmode**(*FileViewMode* m) = 0

Sets the view mode. .

int **index**() = 0

Returns the position of this file view within the panel. .

std::shared\_ptr<*sh::filesystem::FilesystemNode*> **node**()

Returns the current directory.

void **gotoDir**(std::shared\_ptr<*sh::filesystem::FilesystemNode*> node)

Jumps to a new current directory.

Implementations have to call the base class implementation inside.

*sh::filesystem::SizeFormatting* **getSizeFormattingMode**() = 0

Returns the mode how file sizes are formatted for displaying. .

void **setSizeFormattingMode**(*sh::filesystem::SizeFormatting* mode) = 0

Sets the mode how file sizes are formatted for displaying. .

bool **hiddenFilesVisible**() = 0

Returns if hidden files are visible. .

void **setHiddenFilesVisible**(bool v) = 0

Sets the visibility of hidden files. .

int **iconDimension**() = 0

Returns the icon size (in pixels). .

void **setIconDimension**(int v) = 0

Sets the icon size (in pixels). .

void **setSort**(int column, Qt::SortOrder order) = 0

Sets how to sort this view. .

int **sortColumn**() = 0

Returns the index of the current sort column. .

Qt::SortOrder **sortOrder**() = 0

Returns the current sort order. .

QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> **selectedNodes**() = 0

Returns the nodes which the user has selected in this fileview. .

void **setSelection**(const QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> nodes) = 0

Sets the node selection of this fileview. .

QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> **getAllVisibleNodes**() = 0

Returns a list of all nodes currently listed in this file view. .

void **reload** (bool *skipModel* = false)  
 Reloads the data inside this file view.

*sh::filesystem::FilesystemModelFileviewProxy* \***filemodel** ()  
 Returns the filesystem model for this view. .

void **setFilemodel** (*sh::filesystem::FilesystemModelFileviewProxy* \**model*)  
 Sets the filesystem model for this view. .

## Signals

void **selectionChanged** ()  
 Triggered when the selection have changed.

void **viewOptionChanged** ()  
 Triggered when some view options have changed.

## Private Members

*sh::filesystem::FilesystemModelFileviewProxy* \***\_model** = nullptr

std::shared\_ptr<QObject> **\_visibleviewslifetimecanary** = nullptr

std::shared\_ptr<*sh::filesystem::FilesystemNode*> **\_node** = nullptr

**class LogViewDialog** : public *sh::ui::Dialog*  
*#include <logviewdialog.h>* The 'Log' dialog..

Subclassed by *sh::ui::qt::QtLogViewDialog*, *sh::ui::web::WebLogViewDialog*

## Public Functions

**LogViewDialog** (QString *headtext*, QString *subheadtxt*, *sh::base::LogSeverity* *minseverity*)

### Parameters

- *headtext*: The header text.
- *subheadtxt*: The 2nd header text.
- *minseverity*: The minimum message severity this dialog shows initially (the user can typically change that later on).

QString **headtext** ()  
 Returns the header text.

QString **subheadtxt** ()  
 Returns the 2nd header text.

*sh::base::LogSeverity* **minimumSeverity** ()  
 Returns the minimum severity.

qint64 **dialogId** ()  
 Returns the dialog id.  
 Each instance has an id unique in the complete Shallot process lifetime.  
 Must be called in main thread.

bool **isInited** ()  
 Returns if this dialog is initialized.  
 Must be called in main thread.

bool **wasAccepted** ()

Returns if this dialog was ‘accepted’ by the user (typically by clicking some ‘OK’ button).

Must be called in main thread.

void **waitClosed** ()

Wait until the user closed the dialog in some way.

May be called in any thread.

void **close** ()

Closes the dialog.

Must be called in main thread.

bool **wasClosed** ()

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \***manager** ()

Returns the *DialogManager* which hosts this dialog.

## Private Members

QString **\_headtext**

QString **\_subheadtxt**

*sh::base::LogSeverity* **\_minseverity**

**class MainWindow**

*#include <mainwindow.h>* The abstract main window class. There is one instance of it, and it is also used for other ui parts (e.g. creating some dialogs, ...).

Subclassed by *sh::ui::qt::QtMainWindow*, *sh::ui::web::WebMainWindow*

## Public Functions

**MainWindow** ()

**~MainWindow** ()

void **\_initialize** (*sh::base::SingletonInitializer* \**singletonInitializer*)

Initializes the main window. For custom initialization logic, see *MainWindow.initialize*. .

int **fileViewCount** () = 0

Returns the current number of file views. .

void **addFileView** (bool *reinitialize* = false) = 0

Adds a new file view. .

void **removeFileView** (int *i*) = 0

Removes a file view. .

*sh::ui::FileView* \***currentFileView** () = 0

Returns the file view which is currently active (i.e. focussed). .

*sh::ui::FileView* \***getFileView** (int *i*) = 0

Returns a file view by position index. .

void **fileViewsReloadAll** ()

Reload all content in each file view.



```

void onFileViewOptionsChanged (std::function<void> int
    > fcn, QObject *owner = 0) Sets a handler for some view options in a file view changed (optionally
    bound to an owner lifetime).

void onCurrentFileViewChanged (std::function<void>
    > fcn QObject *owner = 0) Sets a handler for the currently active file view changed (optionally bound to
    an owner lifetime).

void onFileViewCountChanged (std::function<void>
    > fcn QObject *owner = 0) Sets a handler for the number of file views changed (optionally bound to an
    owner lifetime).

void onCurrentDirectoryChanged (std::function<void>
    > fcn QObject *owner = 0) Sets a handler for the current directory in the active file view changed (op-
    tionally bound to an owner lifetime).

void onGlobalViewOptionsChanged (std::function<void>
    > fcn QObject *owner = 0) Sets a handler for some global view options changed (optionally bound to an
    owner lifetime).

void onCurrentProfileChanged (std::function<void>
    > fcn QObject *owner = 0) Sets a handler for the current profile changed (optionally bound to an owner
    lifetime).

void jumpToEurl (std::shared_ptr<const sh::filesystem::Eurl> eurl)
    Let the current view jump to another location. .

void jumpToNode (std::shared_ptr<sh::filesystem::FilesystemNode> node) = 0
    Let the current view jump to another location. .

void setTitlePattern (QString value)
    Sets the window title pattern. .

QString titlePattern ()
    Returns the window title pattern.

void setCurrentProfile (QString profile)
    Sets the current profile. .

QString currentProfile ()
    Returns the current profile.

void reloadProfile ()
    Reloads the profile data.

void setTreeVisible (bool v)
    Sets the visibility of the directory tree. .

bool treeVisible ()
    Returns the visibility of the directory tree.

void setTreeSticky (bool v)
    Sets if the directory tree should follow the active file view. .

bool treeSticky ()
    Returns if the directory tree follows the active file view.

void setFileDetailsPanelVisible (bool v)
    Sets the visibility of the file details panel. .

bool fileDetailsPanelVisible ()
    Returns the visibility of the file details panel.

```

std::shared\_ptr<sh::filesystem::FileSystemNode> **currentDirectory** () = 0

Gets the *sh::filesystem::FileSystemNode* selected in the current view. .

std::shared\_ptr<sh::ui::ActionExecutionInfoPanel> **addInfoPanel** (int *position*,  
*sh::actions::ActionExecutionInfo*  
*\*info*, QColor *color* =  
QColor()) = 0

Creates a new action execution panel.

Let this smart pointer die for removing it.

std::shared\_ptr<sh::ui::ActionExecutionInfoPanel> **addInfoPanel** (*sh::actions::ActionExecutionInfo*  
*\*info*, QColor *color* =  
QColor())

Creates a new action execution panel.

Let this smart pointer die for removing it.

void **jumpbarSetTextMode** () = 0

Sets the jumpbar to text input mode. .

void **jumpbarSetButtonMode** () = 0

Sets the jumpbar to button mode. .

bool **jumpbarIsTextMode** () = 0

Returns if the jumpbar is in text input mode. .

QString **toolbarPosition** () = 0

Returns the current toolbar position. .

void **setToolbarPosition** (QString *pos*) = 0

Sets the toolbar position. .

QString **detailPanelPosition** () = 0

Returns the current detail panel position. .

void **setDetailPanelPosition** (QString *pos*) = 0

Sets the detail panel position. .

std::shared\_ptr<AboutDialog> **showAboutDialog** () = 0

Shows and returns an 'About' dialog. .

std::shared\_ptr<ManageProfilesDialog> **showManageProfilesDialog** () = 0

Shows and returns a 'Manage Saved Settings' dialog. .

std::shared\_ptr<OpenWithDialog> **showOpenWithDialog** (std::shared\_ptr<sh::filesystem::FileSystemNode>  
*node*,  
QList<std::shared\_ptr<sh::tools::filetypes::OpenMethod>>  
*openMethods*) = 0

Shows and returns a 'Open With' dialog. .

std::shared\_ptr<StoreProfileDialog> **showStoreProfileDialog** (std::shared\_ptr<const  
*sh::filesystem::Eurl*> *eurl*) =  
0

Shows and returns a 'Save Settings' dialog. .

std::shared\_ptr<TuningDialog> **showTuningDialog** () = 0

Shows and returns a 'Tuning' dialog. .

std::shared\_ptr<LogViewDialog> **showLogViewDialog** (QString *headtext* = QString(),  
QString *subheadtxt* = QString(),  
*sh::base::LogSeverity* *minseverity* =  
*sh::base::LogSeverity::\_DEFAULT*) = 0

Shows and returns a ‘Log’ dialog. .

`std::shared_ptr<ManageBookmarksDialog> showManageBookmarksDialog () = 0`

Shows and returns a ‘Manage Bookmarks’ dialog. .

`std::shared_ptr<FilePropertyDialog> showFilePropertyDialog (QList<std::shared_ptr<sh::filesystem::FilesystemNodes> nodes) = 0`

Shows and returns a ‘File Properties’ dialog. .

`std::shared_ptr<ActionExecutionInfoDialog> createActionExecutionInfoDialog (sh::actions::ActionExecutionInfo *info) = 0`

Creates an action execution dialog. .

`std::shared_ptr<ActionExecutionInfoPanel> showErrorPanel ()`

Shows an error panel.

`std::shared_ptr<DialogManager> dialogManager () = 0`

Returns the *DialogManager* which creates and drives *Dialogs* for this main window.

You typically should not need to use it directly.

`void _closeDirectly ()`

Directly begin application shutdown without checks.

`bool closeApp ()`

Check if the application may shut down now, and if so, initiate it.

`bool isCloseable (QString *msg = nullptr)`

Check if the application may shut down now.

`void handleCloseAppRejected (QString rejectmsg) = 0`

React on a rejected application close request (with some feedback message). .

`void handleClosed ()`

Do some cleanup steps just when closing starts. Implementations must call base implementation!

`bool isClosing ()`

Returns if the application is currently closing.

## Public Static Functions

`MainWindow *mainWindow ()`

Returns the instance of this singleton.

`void setMainWindow (MainWindow *mainWindow)`

Sets the main window instance. .

`bool isReady ()`

Returns if this process ui is ready (i.e. is created or runs headless).

`bool runsHeadless ()`

Returns if this process runs headless, i.e. without any ui, just for a background process.

`void _closeDirectly (sh::base::SingletonInitializer *singletonInitializer)`

`QString uiMode ()`

Returns the UI mode (e.g. qt, web).

### Private Functions

```
void _refreshIsCloseable ()
```

### Private Members

```
bool _closing = false
bool _treeSticky = true
bool _treeVisible = true
bool _detailsPanelVisible = true
QString _titlePattern
QList<std::shared_ptr<sh::ui::ActionExecutionInfoPanel>> _errorpanels
QString _currentProfile
QString _isCloseable
bool _thumbnailrequestongoing = false
bool _thumbnailrequestfollowup = false
int _thumbnailrequestfollowup_width = 0
int _thumbnailrequestfollowup_height = 0
std::function<void ()> _thumbnailrequestfollowup_onBeforeRequest = nullptr
std::function<void (QIcon)> _thumbnailrequestfollowup_onArrived = nullptr
int _thumbnaillastreqwidth = 0
int _thumbnaillastreqheight = 0
quint64 _thumbnaillastrequestid = 0
```

### Private Static Attributes

```
MainWindow *_mainwindow = nullptr
bool _mainwindow_runheadless = false
KillHelperThread *_killthread = nullptr
```

### Friends

```
friend class sh::exceptions::Exception
class KillHelperThread: public QThread
    Helps application shutdown in some situations ;).
```

## Public Functions

void **run** ()

**class ManageBookmarksDialog** : public *sh::ui::Dialog*

*#include <managebookmarksdialog.h>* The ‘Manage Bookmarks’ dialog.

Subclassed by *sh::ui::qt::QtManageBookmarksDialog*, *sh::ui::web::WebManageBookmarksDialog*

## Public Functions

**ManageBookmarksDialog** ()

qint64 **dialogId** ()

Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

bool **isInitiated** ()

Returns if this dialog is initialized.

Must be called in main thread.

bool **wasAccepted** ()

Returns if this dialog was ‘accepted’ by the user (typically by clicking some ‘OK’ button).

Must be called in main thread.

void **waitClosed** ()

Wait until the user closed the dialog in some way.

May be called in any thread.

void **close** ()

Closes the dialog.

Must be called in main thread.

bool **wasClosed** ()

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \***manager** ()

Returns the *DialogManager* which hosts this dialog.

**class ManageProfilesDialog** : public *sh::ui::Dialog*

*#include <manageprofilesdialog.h>* The ‘Manage Saved Settings’ dialog.

Subclassed by *sh::ui::qt::QtManageProfilesDialog*, *sh::ui::web::WebManageProfilesDialog*

## Public Functions

**ManageProfilesDialog** ()

qint64 **dialogId** ()

Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

bool **isInit** ()

Returns if this dialog is initialized.

Must be called in main thread.

bool **wasAccepted** ()

Returns if this dialog was 'accepted' by the user (typically by clicking some 'OK' button).

Must be called in main thread.

void **waitClosed** ()

Wait until the user closed the dialog in some way.

May be called in any thread.

void **close** ()

Closes the dialog.

Must be called in main thread.

bool **wasClosed** ()

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \***manager** ()

Returns the *DialogManager* which hosts this dialog.

**class OpenWithDialog** : public *sh::ui::Dialog*

#include <openwithdialog.h> The 'Open With' dialog.

Subclassed by *sh::ui::qt::QtOpenWithDialog*, *sh::ui::web::WebOpenWithDialog*

## Public Functions

**OpenWithDialog** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> *node*,  
QList<std::shared\_ptr<*sh::tools::filetypes::OpenMethod*>> *openMethods*)

### Parameters

- *node*: The file which is later to be opened.
- *openMethods*: The open-methods (programs for opening the file) to present for choice.

std::shared\_ptr<*sh::tools::filetypes::OpenMethod*> **chosenMethod** () = 0

Returns the chosen open-method (program for opening the file).

bool **rememberForMimetype** () = 0

Returns if the chosen method is checked to be remembered for the same mime-type in the future.

std::shared\_ptr<const *sh::filesystem::Eurl*> **rememberForDirectory** () = 0

If it's checked for the chosen method to be remembered for some subdirectory in the future, it returns the Eurl of this subdirectory, otherwise `nullptr`.

bool **rememberForFile** () = 0

Returns if the chosen method is checked to be remembered for the same file in the future.

std::shared\_ptr<sh::filesystem::FilesystemNode> **node** ()

Returns the file node which is later to be opened.

QList<std::shared\_ptr<tools::filetypes::OpenMethod>> **openMethods** ()

Returns the open-methods (programs for opening the file) to present for choice.

qint64 **dialogId** ()

Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

bool **isInited** ()

Returns if this dialog is initialized.

Must be called in main thread.

bool **wasAccepted** ()

Returns if this dialog was 'accepted' by the user (typically by clicking some 'OK' button).

Must be called in main thread.

void **waitClosed** ()

Wait until the user closed the dialog in some way.

May be called in any thread.

void **close** ()

Closes the dialog.

Must be called in main thread.

bool **wasClosed** ()

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \***manager** ()

Returns the *DialogManager* which hosts this dialog.

## Private Members

std::shared\_ptr<sh::filesystem::FilesystemNode> **\_node**

QList<std::shared\_ptr<sh::tools::filetypes::OpenMethod>> **\_openMethods**

**class SearchPanelAbstractEditor**

#include <searchpanelconfiguration.h> Abstract base class for editor widgets in a search panel.

Subclassed by *sh::ui::SearchPanelDateTimeEditor*, *sh::ui::SearchPanelLabelEditor*,  
*sh::ui::SearchPanelSpacerEditor*, *sh::ui::SearchPanelTextEditor*

## Public Functions

**SearchPanelAbstractEditor** () = default

**~SearchPanelAbstractEditor** () = default

bool **isWidgetEnabled** () = 0  
Returns if the widget is enabled. .

void **setWidgetEnabled** (bool v) = 0  
Sets if the widget is enabled. .

**class SearchPanelButton**

*#include <searchpanelconfiguration.h>* A button in the button bar of a search panel.

Subclassed by *sh::ui::qt::QtSearchPanelButton*

## Public Functions

**SearchPanelButton** ()

**~SearchPanelButton** ()

void **setButtonText** (QString txt) = 0  
Sets the button text. .

void **setMenuSelection** (int i) = 0  
Sets the currently selected menu item (for buttons with menus). .

**class SearchPanelConfiguration**

*#include <searchpanelconfiguration.h>* Configuration for a search panel population.

It is populated with ui controls by the chosen search criterion. Later on those controls are used for updating the configuration data.

Subclassed by *sh::ui::qt::QtSearchPanelConfiguration*

## Public Functions

**SearchPanelConfiguration** ()

**~SearchPanelConfiguration** ()

*SearchPanelButton* \***addMenuButton** (QString text, QStringList menu, std::function<void> int  
> onChanged = 0) Adds and returns a button for a menu. .

*SearchPanelButton* \***addActionButton** (QString text, std::function<void>  
> action = 0) Adds and returns a button for an action. .

*SearchPanelTextEditor* \***addTextEditor** () = 0  
Adds and returns a text editor. .

*SearchPanelDateTimeEditor* \***addDateTimeEditor** () = 0  
Adds and returns a date/time editor. .

*SearchPanelLabelEditor* \***addLabel** () = 0  
Adds and returns a label. .

*SearchPanelSpacerEditor* \***addSpacer** () = 0  
Adds and returns a spacer. .



*SearchPanelAbstractEditor* \*getEditorAt (int i) = 0

Returns the editor widget at a given position. .

void onDestroyed (std::function<void>)

> fctQObject \*owner = 0 Sets a handler for panel removal (optionally bound to an owner lifetime).

## Public Members

*\_SearchPanelConfiguration\_HelperQObject* myqobject

**class SearchPanelDateTimeEditor : public *sh::ui::SearchPanelAbstractEditor***

*#include <searchpanelconfiguration.h>* A date/time picker for search panel usage.

Subclassed by *sh::ui::qt::QtSearchPanelAbstractEditor< QDateTimeEdit, SearchPanelDateTimeEditor >*

## Public Functions

**SearchPanelDateTimeEditor** () = default

QDateTime **datetime** () = 0

Returns the selected date/time. .

void **setDatetime** (QDateTime s) = 0

Sets the selected date/time.

bool **isWidgetEnabled** () = 0

Returns if the widget is enabled. .

void **setWidgetEnabled** (bool v) = 0

Sets if the widget is enabled. .

**class SearchPanelLabelEditor : public *sh::ui::SearchPanelAbstractEditor***

*#include <searchpanelconfiguration.h>* A text label for search panel usage.

Subclassed by *sh::ui::qt::QtSearchPanelAbstractEditor< QLabel, sh::ui::SearchPanelLabelEditor >*

## Public Functions

**SearchPanelLabelEditor** () = default

QString **textContent** () = 0

Returns the text content. .

void **setTextContent** (QString s) = 0

Sets the text content. .

*SearchPanelAbstractEditor* \*focusProxyEditor () = 0

Returns the focus proxy widget (which focus gets redirected to in some situations). .

void **setFocusProxyEditor** (*SearchPanelAbstractEditor* \*w) = 0

Sets the focus proxy widget (which focus gets redirected to in some situations). .

bool **isWidgetEnabled** () = 0

Returns if the widget is enabled. .

void **setWidgetEnabled** (bool v) = 0

Sets if the widget is enabled. .

```
class SearchPanelSpacerEditor : public sh::ui::SearchPanelAbstractEditor
    #include <searchpanelconfiguration.h> A spacer for search panel usage.

    Subclassed by sh::ui::qt::QtSearchPanelAbstractEditor< QLabel, sh::ui::SearchPanelSpacerEditor >
```

### Public Functions

**SearchPanelSpacerEditor** () = default

bool **isWidgetEnabled** () = 0  
Returns if the widget is enabled. .

void **setWidgetEnabled** (bool v) = 0  
Sets if the widget is enabled. .

```
class SearchPanelTextEditor : public sh::ui::SearchPanelAbstractEditor
    #include <searchpanelconfiguration.h> A single line text editor for search panel usage.

    Subclassed by sh::ui::qt::QtSearchPanelAbstractEditor< QLineEdit, sh::ui::SearchPanelTextEditor >
```

### Public Functions

**SearchPanelTextEditor** () = default

QString **textContent** () = 0  
Returns the text content. .

void **setTextContent** (QString s) = 0  
Sets the text content. .

QString **placeholderDescription** () = 0  
Returns the placeholder description text (visible if field is empty). .

void **setPlaceholderDescription** (QString s) = 0  
Sets the placeholder description text (visible if field is empty). .

bool **isWidgetEnabled** () = 0  
Returns if the widget is enabled. .

void **setWidgetEnabled** (bool v) = 0  
Sets if the widget is enabled. .

```
class StoreProfileDialog : public sh::ui::Dialog
    #include <storeprofiledialog.h> The 'Save Settings' dialog.

    Subclassed by sh::ui::qt::QtStoreProfileDialog, sh::ui::web::WebStoreProfileDialog
```

### Public Functions

**StoreProfileDialog** (std::shared\_ptr<const *sh::filesystem::Eurl*> eurl)  
**Parameters**

- eurl: The current directory this dialog shall work on.

std::shared\_ptr<const *sh::filesystem::Eurl*> **eurl** ()  
Returns the current directory this dialog shall work on.

QList<*SettingEntry*> **checkedSettings** () = 0  
Returns the list of settings the user has checked to store.

QString **profileName** () = 0

Returns the profile the user has chosen to store settings for.

bool **withSubDirectories** () = 0

Returns if the user has chosen to apply the checked settings also for subdirectories.

QStringList **inheritsFrom** () = 0

Returns the list of profiles the user has chosen to inherit from.

bool **hasGlobal** () = 0

Returns if the user has chosen to apply the checked settings for each directory (instead of the current directory).

qint64 **dialogId** ()

Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

bool **isInitd** ()

Returns if this dialog is initialized.

Must be called in main thread.

bool **wasAccepted** ()

Returns if this dialog was 'accepted' by the user (typically by clicking some 'OK' button).

Must be called in main thread.

void **waitClosed** ()

Wait until the user closed the dialog in some way.

May be called in any thread.

void **close** ()

Closes the dialog.

Must be called in main thread.

bool **wasClosed** ()

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \***manager** ()

Returns the *DialogManager* which hosts this dialog.

## Private Members

std::shared\_ptr<const *sh::filesystem::Eurl*> **\_eurl**

**class SettingEntry**

*#include <storeprofiledialog.h>* A storable entry in a 'Save Settings' dialog.

## Public Functions

**SettingEntry** (*sh::settings::Setting* \**setting*, int *onlyInFileview* = -1)

*sh::settings::Setting* \***setting** ()

Returns the setting to store.

int **fileview** ()

Returns the index of the file view to store this setting for (or -1 for all).

bool **isForFileview** ()

Returns if this setting is to be stored for a particular file view (instead of for all).

## Private Members

*sh::settings::Setting* \*\_**setting**

int **\_fileview**

**class TuningDialog** : public *sh::ui::Dialog*

*#include <tuningdialog.h>* The ‘Tuning’ dialog.

Subclassed by *sh::ui::qt::QtTuningDialog*, *sh::ui::web::WebTuningDialog*

## Public Functions

**TuningDialog** ()

qint64 **dialogId** ()

Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

bool **isInitied** ()

Returns if this dialog is initialized.

Must be called in main thread.

bool **wasAccepted** ()

Returns if this dialog was ‘accepted’ by the user (typically by clicking some ‘OK’ button).

Must be called in main thread.

void **waitClosed** ()

Wait until the user closed the dialog in some way.

May be called in any thread.

void **close** ()

Closes the dialog.

Must be called in main thread.

bool **wasClosed** ()

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \***manager** ()

Returns the *DialogManager* which hosts this dialog.

**namespace qt**

The Qt user interface implementation.

This is the default UI. It builds a desktop application based on the Qt gui components.

**class QLineEditWithKeyboardShortcuts : public QLineEdit**  
*#include <qtjumpbar.h>* Helper widget of *QtJumpBar*.

**Public Functions**

**LineEditWithKeyboardShortcuts** (QWidget \*parent = 0)

**Signals**

void **enterPressed** ()

void **escapePressed** ()

**class QtAboutDialog : public sh::ui::qt::QtDialog, public sh::ui::AboutDialog**  
*#include <qtaboutdialog.h>* Qt based about dialog.

**Public Functions**

**QtAboutDialog** ()

**~QtAboutDialog** ()

qint64 **dialogId** ()

Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

bool **isInitied** ()

Returns if this dialog is initialized.

Must be called in main thread.

bool **wasAccepted** ()

Returns if this dialog was 'accepted' by the user (typically by clicking some 'OK' button).

Must be called in main thread.

void **waitClosed** ()

Wait until the user closed the dialog in some way.

May be called in any thread.

void **close** ()

Closes the dialog.

Must be called in main thread.

bool **wasClosed** ()

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \***manager** ()

Returns the *DialogManager* which hosts this dialog.

## Private Members

*Ui::QtAboutDialog* \*ui

## Private Slots

void on\_btnClose\_clicked()

void on\_btnLicense\_clicked()

void on\_btnHomepage\_clicked()

**class** **QtActionExecutionInfoDialog**: public QDialog, public *sh::ui::ActionExecutionInfoDialog*  
#include <qtactionexecutioninfodialog.h> Qt progress dialog for action executions.

## Public Types

**enum** **MessageBoxButton**

Buttons in a message box from *ActionExecutionUserFeedback*.

*Values:*

**enumerator** NONE = 0

**enumerator** OK = 1 << 0

**enumerator** Continue = 1 << 1

**enumerator** Cancel = 1 << 2

**enumerator** Retry = 1 << 3

**enumerator** Yes = 1 << 4

**enumerator** No = 1 << 5

## Public Functions

**QtActionExecutionInfoDialog** (*sh::actions::ActionExecutionInfo* \*info, QWidget \*parent = 0)

**~QtActionExecutionInfoDialog**()

void **setDetails** (QString fv, QString fob, QString tv, QString tob)

Sets the item details text ('from a/foo.jpg', 'to b/foo.jpg'). .

void **setHead** (QString txt)

Sets the header text. .

void **setProgress** (bool isDeterminate, quint64 done, quint64 all, QString text)

Sets the progress. .

int **messageBox** (QString text, QList<QString> answers, QString icon = QString(), int defaultanswer = -1, int cancelanswer = -1, QList<QString> answericons = QList<QString>())

int **inputBox** (QString text, QList<QString> answers, QString \*value, QString icon = QString(), int defaultanswer = -1, int cancelanswer = -1, int valuePreselectFrom = -1, int valuePreselectTo = -1)

int **multilineInputBox** (QString text, QList<QString> answers, QString \*value, QString icon = QString(), int defaultanswer = -1, int cancelanswer = -1)

```

int simpleChooserGridform (QString text, GridformEntries *entries, QStringList answers,
                           int defaultanswer = -1, int cancelanswer = -1)

bool credentialsDialog (QString text, bool showDomain, bool showUsername, bool
                        showPassword, bool showAnonymous, bool showRemember,
                        QString *domain, QString *username, QString *password, bool
                        *isAnonymous, bool *isRemember)

bool unixPermissionsDialog (bool *userMayRead, bool *userMayWrite, bool *user-
                            MayExecute, bool *groupMayRead, bool *groupMay-
                            Write, bool *groupMayExecute, bool *othersMayRead,
                            bool *othersMayWrite, bool *othersMayExecute, bool
                            *sticky, bool *setuid, bool *setgid, QStringList users,
                            QStringList groups, QString *ownerUser, QString *own-
                            erGroup)

void setLogicallyVisible (bool v)
    Sets if this dialog is logically visible (i.e. set visible by the action). .

void setBackground (bool v)
    Sets if this dialog is currently in background mode (i.e. not visible). .

void setForceForeground (bool v)
    Sets if this dialog is currently forced to be visible in foreground. .

bool isLogicallyVisible ()
    Returns if this dialog is logically visible (i.e. set visible by the action).

bool isBackground ()
    Returns if this dialog is currently in background mode (i.e. not visible).

QString simpleInputDialog (QString text, QString deflt, int valuePreselectFrom = -1, int val-
                           uePreselectTo = -1)

int simpleMessageBox (QString text, int buttons = (int) MessageBoxButton::OK, QString
                     icon = QString(), int defaultbutton = (MessageBoxButton)0, int can-
                     celbutton = (MessageBoxButton)0)

```

## Private Functions

```
void _computevisibility ()
```

## Private Members

```
Ui::QtActionExecutionInfoDialog *ui
```

```
sh::ui::qt::feedbackpanels::FeedbackPanel *currentFeedbackPanel = 0
```

### Private Slots

```
void on_btnCancel_clicked()
void on_btnBackground_clicked()
```

```
class QtActionExecutionInfoPanel : public QWidget, public sh::ui::ActionExecutionInfoPanel
#include <qtactionexecutioninfopanel.h> Qt status bar info-panel for an action execution.
```

### Public Functions

```
QtActionExecutionInfoPanel (sh::actions::ActionExecutionInfo *info, QColor color,
                             QWidget *parent = 0)

void setLabel (QString s)
    Sets the label text. .

void setWidth (int width)
    Sets the panel with (in pixels). .

QSize sizeHint () const

void setProgress (bool isProgressDeterminate, quint64 progressDone, quint64 progressAll)
    Sets the progress. .

void setForceForeground (bool v)
    Sets if the associated action is currently forced to be visible in foreground. .

void setPanelVisible (bool v)
    Sets if the panel is visible. .

void onClicked (std::function<void>
    > fctQObject *owner = 0) Sets a handler for a click on the button (optionally bound to an owner
    lifetime).

void onDestroyed (std::function<void>
    > fctQObject *owner = 0) Sets a handler for panel removal (optionally bound to an owner lifetime).

void onVisibilityChanged (std::function<void>
    > fctQObject *owner = 0) Sets a handler for panel visibility changes (optionally bound to an owner
    lifetime).
```

### Private Functions

```
void _check_indeterminateanimationtimer_enable()
```

### Private Members

```
QString lbl1
bool _progressDeterminate = false
int _indeterminateAnimationCounter = 0
int _width
QTimer *_indeterminateAnimationTimer
quint64 _progressAll
```



```

quint64 _progressDone
bool _isforeground_forced = false
sh::actions::ActionExecutionInfo *info
QPixmap gradient
QColor colorProgress2
QColor colorProgress1
QColor colorBase
QColor colorFrame
QColor colorText
QColor colorTextDark
class QActionMenu : public QMenu
    #include <qtactionmenu.h> Qt based action menu used for context menus and in the toolbar.

```

### Public Functions

```

QActionMenu (QWidget *parent = 0)
void setHeader (QString t)
QList<QAction*> allActions ()
void clearAllActions ()
void removeActionAt (int i)
QAction *addNewAction (int i = -1)
QAction *addNewSubMenu (sh::ui::qt::QActionMenu **qsubmenu, int i = -1)
QAction *addNewSeparator (int i = -1)
QAction *addNewHeader (int i = -1)
bool actionIsHeader (QAction *a)

```

### Private Functions

```

void _observeAction (QAction *a)
void _correctMenuPosition ()

```

### Private Members

```

QColor brandingcolor
QFont _boldfont
QFont _normalfont
int _origMenuPosX = -1
int _origMenuPosY = -1
QString _header

```

```
int _cntInternalActions

class QActionMenuHandler
    #include <qtactionmenuhandler.h> A handler which reflects the sh::actions objects to a graphical
    menu (and keeps that up-to-date).
```

### Public Functions

```
QActionMenuHandler (std::shared_ptr<sh::actions::ActionInstantiation> ai,
                    std::shared_ptr<QtActionMenu> menu)
```

### Private Functions

```
void _markDefault2 ()
void _append_actions (sh::ui::qt::QtActionMenu *menu, QList<std::shared_ptr<sh::actions::ActionInstantiation>>
                     acts, std::shared_ptr<sh::actions::ActionCategory> category)
```

### Private Members

```
QList<std::shared_ptr<sh::actions::ActionInstantiation>> selects
QList<std::shared_ptr<sh::actions::ActionInstantiation>> directs
QHash<QAction*, std::shared_ptr<sh::actions::AbstractActionItem>> qaction2action
std::weak_ptr<QtActionMenu> menu
```

### Private Static Functions

```
std::shared_ptr<sh::actions::ActionActionItem> _getDefaultAction (QList<std::shared_ptr<sh::actions::AbstractActionItem>>
                                                                actionList)
void _markDefault (std::weak_ptr<sh::actions::SubmenuItem> itmSubmenu,
                  sh::ui::qt::QtActionMenu *menu)
QAction *_createAndConnectAction (sh::actions::AbstractActionItem *itm,
                                  sh::ui::qt::QtActionMenu *menu,
                                  std::function<void>
                                  > onChangedQObject *onchangedbuddy = 0)
void _applyPropertiesToQAction (sh::actions::AbstractActionItem *itm, QAction
                               *_widgetaction)
void _updateSubmenu (sh::actions::SubmenuItem *itm, QtActionMenu *menu,
                    std::function<void>
                    > onChanged = 0)
```

## Friends

**friend class** QtToolBarButtonHandler

**class** QtDialog : public QDialog

*#include <qtdialog.h>* Abstract base class for a qt based dialog. Typically used for also implementing some *sh::ui::Dialog*.

Subclassed by *sh::ui::qt::QtAboutDialog*, *sh::ui::qt::QtExceptionDialog*,  
*sh::ui::qt::QtFilePropertyDialog*, *sh::ui::qt::QtLogViewDialog*, *sh::ui::qt::QtManageBookmarksDialog*,  
*sh::ui::qt::QtManageProfilesDialog*, *sh::ui::qt::QtOpenWithDialog*,  
*sh::ui::qt::QtStoreProfileDialog*, *sh::ui::qt::QtTuningDialog*

## Public Functions

QtDialog()

**class** QtDialogManager : public *sh::ui::DialogManager*

*#include <qtdialog.h>* A *DialogManager* used in Qt ui.

## Public Functions

std::shared\_ptr<*Dialog*> **getDialogById** (qint64 *id*)

Returns a *Dialog* by dialog id.

Must be called in main thread.

QList<std::shared\_ptr<*Dialog*>> **getAllDialogs** ()

Returns a list of all dialogs currently shown (in order of creation).

Must be called in main thread.

QList<qint64> **getAllDialogIds** ()

Returns a list of dialog ids of all dialogs currently shown (in order of creation).

Must be called in main thread.

template<class **TDlg**, typename ...**Args**>

std::shared\_ptr<*TDlg*> **createAndShowDialog** (*Args*... *args*)

Creates and shows a *Dialog* by class and constructor parameters.

Those dialogs (*TDlg*) have to implement *Dialog* (typically a subclass of it, representing a particular dialog, like *FilePropertyDialog*), and also some ui mode (e.g. qt, web) specific class (depends on that ui mode).

You typically should not have to use it outside of *MainWindow* or subclasses. The *MainWindow* interface allows to create all available dialogs in a ui mode independent way.

May be called in any thread.

**Return** The created *Dialog* instance.

## Private Functions

void **showDialog** (std::shared\_ptr<*Dialog*> *dialog*) **override**

This method implements the ui mode specific mechanism for showing a dialog.

Must be called in main thread.

**class QtExceptionHandler**: public *sh::ui::qt::QtDialog*, public *sh::ui::ExceptionHandler*  
#include <qtexceptiondialog.h> Qt based exception dialog.

## Public Functions

**QtExceptionHandler** (QString *error1*, QString *error2*, QString *details*, QString *icon*, bool  
*mayRetry*, bool *mayClose*, bool *mayCancel*, bool *showLoglabelAnd-*  
*Details*)

**~QtExceptionHandler** ()

*ExceptionHandlerResult* **answer** () **override**

Returns the answer the user has given in this dialog.

QString **error1** ()

Returns the 1st error text.

QString **error2** ()

Returns the 2nd error text.

QString **details** ()

Returns the error details.

QString **icon** ()

Returns the icon (as name resolveable by *sh::base::IconManager*).

bool **mayRetry** ()

Returns if it's allowed to retry.

bool **mayClose** ()

Returns if it's allowed to just close and continue without closing Shallot.

bool **mayCancel** ()

Returns if it's allowed to cancel, i.e. throwing a *sh::exceptions::CancelException*.

bool **showLoglabelAndDetails** ()

Returns if the dialog shall allow to read the details.

qint64 **dialogId** ()

Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

bool **isInited** ()

Returns if this dialog is initialized.

Must be called in main thread.

bool **wasAccepted** ()

Returns if this dialog was 'accepted' by the user (typically by clicking some 'OK' button).

Must be called in main thread.

void **waitClosed** ()  
 Wait until the user closed the dialog in some way.  
 May be called in any thread.

void **close** ()  
 Closes the dialog.  
 Must be called in main thread.

bool **wasClosed** ()  
 Returns if this dialog was closed.  
 Must be called in main thread.

*DialogManager* \***manager** ()  
 Returns the *DialogManager* which hosts this dialog.

### Private Members

*Ui::QtExceptionDialog* \***ui**  
*ExceptionDialogResult* **\_answer** = *ExceptionDialogResult::Shutdown*

### Private Slots

void **on\_btnExit\_clicked** ()  
 void **on\_btnClose\_clicked** ()  
 void **on\_btnCancel\_clicked** ()  
 void **on\_btnRetry\_clicked** ()  
 void **on\_btnShowLog\_clicked** ()  
 void **on\_btnExitAndSaveLog\_clicked** ()  
 void **on\_btnShowDetails\_toggled** (bool *checked*)

**class QtFileDetailsPanel : public QWidget**  
*#include <qtfiledetailspanel.h>* The details panel.  
 Can be shown in the main window. It presents detail information about the selected file.

### Public Functions

**QtFileDetailsPanel** (QWidget \**parent* = 0)  
 void **setNodes** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> *nodes*)  
 void **setOrientation** (Qt::Orientation *orientation*)  
 QSize **sizeHint** () **const override**  
 ~**QtFileDetailsPanel** ()

### Private Members

```
int PADDINGX
int PADDINGY
QList<std::shared_ptr<sh::filesystem::FilesystemNode>> _nodes
QList<std::shared_ptr<sh::paneldetails::PanelDetail>> _panelDetails
QList<DetailPlacement> _placements
QFont fontNormal
QFont fontBold
QPixmap _widgetimagecache
QTimer _widgetimagecachetimer
Qt::Orientation _orientation = Qt::Horizontal
class DetailPlacement
    #include <qfiledetailspanel.h>
```

### Public Functions

```
DetailPlacement (int x, int y, int w, int h, QList<DetailRowPlacement> rowplacements,
                  std::shared_ptr<sh::paneldetails::PanelDetail> detail)
```

### Public Members

```
std::shared_ptr<sh::paneldetails::PanelDetail> detail
int x
int y
int w
int h
QList<DetailRowPlacement> rowplacements
class DetailRowPlacement
    #include <qfiledetailspanel.h>
```

### Public Functions

```
DetailRowPlacement (int h, int contentx, int contenty, QList<int> elementwidths)
DetailRowPlacement () = default
DetailRowPlacement (const DetailRowPlacement&) = default
```

## Public Members

```

int h
int contentx
int contenty
QList<int> elementwidths

```

```

class QtFileIconview: public QListView, public sh::ui::qt::QtFileView
    #include <qtfileiconview.h> Icon view for the contents of one directory.

```

## Public Functions

```

QtFileIconview (QWidget *parent = 0)
    Constructed only by the infrastructure and made available otherwise.

void setThumbDimension (double size)
    Sets the size of the thumbnail image.

void setBackgroundColor (QString c)
QString backgroundColor ()

QList<std::shared_ptr<sh::filesystem::FilesystemNode>> selectedNodes ()

void setSelection (const QList<std::shared_ptr<sh::filesystem::FilesystemNode>>
                  nodes)

std::shared_ptr<sh::filesystem::FilesystemNode> node ()

QList<std::shared_ptr<sh::filesystem::FilesystemNode>> getAllVisibleNodes ()

void setSizeFormatting (sh::filesystem::SizeFormatting v)

void gotoDir (std::shared_ptr<sh::filesystem::FilesystemNode> n)

void setHiddenFilesVisible (bool value)

QThread *thread ()

void configure (sh::ui::qt::QtFileViewControl *viewctrl)

void focus ()

void setSort (int column, Qt::SortOrder order)

QObject *as_qobject ()

QWidget *as_qwidget ()

QAbstractItemView *as_qabstractitemview ()

sh::ui::qt::QtFileViewControl *control ()

```

### Private Members

```
int _deleg_w = 0
int _deleg_h = 0
int _dsx = 0
int _dsy = 0
int _lineheight = 0
```

```
class QtFileList : public QTreeView, public sh::ui::qt::QtFileView
#include <qtfilelist.h> List view for the contents of one directory.
```

### Public Functions

```
QtFileList (QWidget *parent = 0)
    Constructed only by the infrastructure and made available otherwise.

void setSort (int column, Qt::SortOrder order) override
void setBackgroundColor (QString c)
QString backgroundColor ()
QList<std::shared_ptr<sh::filesystem::FilesystemNode>> selectedNodes ()
void setSelection (const QList<std::shared_ptr<sh::filesystem::FilesystemNode>>
                  nodes)
std::shared_ptr<sh::filesystem::FilesystemNode> node ()
QList<std::shared_ptr<sh::filesystem::FilesystemNode>> getAllVisibleNodes ()
void setSizeFormatting (sh::filesystem::SizeFormatting v)
void gotoDir (std::shared_ptr<sh::filesystem::FilesystemNode> n)
void setHiddenFilesVisible (bool value)
QThread *thread ()
void configure (sh::ui::qt::QtFileViewControl *viewctrl)
void focus ()
QObject *as_qobject ()
QWidget *as_qwidget ()
QAbstractItemView *as_qabstractitemview ()
sh::ui::qt::QtFileViewControl *control ()
```



## Private Functions

QString **getColumnnameForIndex** (int *index*)

void **\_adaptColumnWidth** (int *index*)

## Private Members

bool **\_skip\_slot\_sectionResized** = false

## Private Slots

void **slot\_sectionResized** (int *index*, int *oldsize*, int *newsize*)

## Private Static Attributes

std::shared\_ptr<sh::configuration::ConfigurationValue> **cfgvalFileListIconSize** = sh::configuration::Configur

**class QtFilePropertyDialog**: public sh::ui::qt::QtDialog, public sh::ui::FilePropertyDialog  
*#include <qtfilepropertydialog.h>* Qt based properties dialog.

## Public Functions

**~QtFilePropertyDialog** ()

**QtFilePropertyDialog** (QList<std::shared\_ptr<sh::filesystem::FilesystemNode>> *nodes*)

Constructed only by the infrastructure and made available otherwise.

void **init** (std::shared\_ptr<Dialog> *dialog*) **override**

Executes custom stuff on initialization, e.g. for populating the dialog.

void **refresh** () **override**

Refreshes the dialog content.

sh::ui::FilePropertyDialogTabTableView \***createTabViewTable** () **override**

Creates a new tab view table sub-widget.

sh::ui::FilePropertyDialogTabTextView \***createTabViewText** () **override**

Creates a new tab view text sub-widget.

sh::ui::FilePropertyDialogTabIconTextBannerView \***createTabViewIconTextBanner** ()

**override**

Creates a new tab view icon text banner sub-widget.

QList<std::shared\_ptr<FilePropertyDialogTab>> **tabs** ()

Returns a list of all tabs.

sh::ui::FilePropertyDialogTabActionsView \***widgetAt** (std::shared\_ptr<FilePropertyDialogTab>  
*tab*, int *i*)

Returns the widget from a tab at a certain index.

int **widgetCount** (std::shared\_ptr<FilePropertyDialogTab> *tab*)

Returns the number of widgets in a tab.

qint64 **dialogId** ()

Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

bool **isInitiated**()

Returns if this dialog is initialized.

Must be called in main thread.

bool **wasAccepted**()

Returns if this dialog was 'accepted' by the user (typically by clicking some 'OK' button).

Must be called in main thread.

void **waitClosed**()

Wait until the user closed the dialog in some way.

May be called in any thread.

void **close**()

Closes the dialog.

Must be called in main thread.

bool **wasClosed**()

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \***manager**()

Returns the *DialogManager* which hosts this dialog.

## Public Slots

void **on\_btnClose\_clicked**()

void **on\_btnRefresh\_clicked**()

void **on\_tabWidget\_currentChanged**(int *index*)

void **selectTabByScrollPosition**()

## Public Static Functions

void **addTabFactory**(int *i*, std::shared\_ptr<*FilePropertyDialogTabFactory*> *factory*)

Adds a *FilePropertyDialogTabFactory* so the Properties dialogs show its content.

See also the REGISTER\_FILEPROPERTYDIALOGTAB macro.

### Parameters

- *i*: An index which controls the display order. Use one of the REGISTER\_FILEPROPERTYDIALOGTAB\_INDEX\_\* values in *FilePropertyDialogTabFactory* as base values.

## Private Members

```

Ui::QtFilePropertyDialog *ui
QList<QWidget*> _internaltabwidgets
QList<QString> _tabheads
int _skip_on_tabWidget_currentChanged = 0
int _skip_selectTabByScrollPosition = 0
QVBoxLayout *scrollAreaLayout

```

## Friends

```
friend class FilePropertyDialogTab
```

```

class QtFilePropertyDialogTabActionsView : public QWidget, public sh::ui::FilePropertyDialogTabA
#include <qtfilepropertydialogtabactionsview.h> A tab view which shows a main widget in the main
part and some buttons below.

```

## Public Functions

```
QtFilePropertyDialogTabActionsView (QWidget *parent = 0)
```

Is intended to be directly constructed from everywhere.

```
void setContent (FilePropertyDialogTabViewContent *w)
```

Sets the main widget. .

```
void setButtons (QList<QString> buttons)
```

Sets the list of buttons. .

```
void setVisible (bool v)
```

Sets the view visible or hidden. .

```
FilePropertyDialogTabViewContent *content ()
```

The main widget.

```
QList<QString> buttons ()
```

The list of buttons.

```
void onButtonTriggered (std::function<void> int i
```

> fct, QObject \*owner = 0) Sets a handler for a click on one of the buttons (optionally bound to an owner lifetime).

```

class QtFilePropertyDialogTabIconTextBannerView : public QWidget, public sh::ui::FilePropertyD
#include <qtfilepropertydialogtabicontextbannerview.h> Qt based FilePropertyDialogTabIcon-
TextBannerView.

```

### Public Functions

**QtFilePropertyDialogTabIconTextBannerView** (QWidget *\*parent* = 0)

void **clear** ()

Clears the contents.

void **insertIcon** (QIcon *icon*, int *sizePt* = 24)

Adds an icon.

void **insertText** (QString *text*)

Adds a text.

### Private Members

QHBoxLayout *\*\_layout*

**class QtFilePropertyDialogTabTableView** : public QTableView, public *sh::ui::FilePropertyDialogTabTableView*  
*#include <qtfilepropertydialogtabtableview.h>* Qt based *FilePropertyDialogTabTableView*.

### Public Types

**typedef** QPair<int, int> **ItemIndex**

### Public Functions

**QtFilePropertyDialogTabTableView** (QWidget *\*parent* = 0)

void **setContent** (QList<QPair<QString, QString>> *content*)

Sets the content.

QList<*ItemIndex*> **getSelectedItems** ()

Returns the selected cells.

QVariant **getItemValue** (int *r*, int *c*)

Returns a value of a cell.

### Private Functions

void **createAndSetModel** ()

**class QtFilePropertyDialogTabTextView** : public QLabel, public *sh::ui::FilePropertyDialogTabTextView*  
*#include <qtfilepropertydialogtabtextview.h>* Qt based *FilePropertyDialogTabTextView*.

### Public Functions

**QtFilePropertyDialogTabTextView** (QWidget *\*parent* = 0)

void **setContent** (QString *content*)

Sets the content.

**class QtFilesystemPanel** : public QWidget  
*#include <qtfilesystempanel.h>* A splitted horizontal panel of file views.

## Public Functions

```

QtFilesystemPanel (QWidget *parent = 0)
~QtFilesystemPanel ()
void addView (bool reinitialize = false)
void removeView (int i)
sh::ui::FileView *currentView ()
int currentViewIndex ()
int viewsCount ()
sh::ui::FileView *view (int i)
void selectFolder (std::shared_ptr<sh::filesystem::FilesystemNode> folder, int index = -1)
QList<std::shared_ptr<sh::filesystem::FilesystemNode>> selectedNodes ()
void _emit_viewmodeChanged (int i)
void buildView (std::shared_ptr<sh::ui::qt::QtFileView> *list, sh::ui::qt::QtFileViewControl
                *viewctrl, bool isNew)
void setCustomWidget (int i, std::shared_ptr<QWidget> w)
std::shared_ptr<sh::ui::qt::QtFileView> viewwidget (int i)

```

## Signals

```

void panelFolderActivated (std::shared_ptr<sh::filesystem::FilesystemNode> folder,
                           bool realSwitch)
void panelCurrentViewChanged ()
void panelViewmodeChanged (int i)
void panelViewOptionChanged (int i)
void panelViewCountChanged ()
void panelSelectionChanged ()

```

## Private Members

```

Ui::QtFilesystemPanel *ui
std::shared_ptr<sh::ui::qt::QtFileView> activeView
QString activeColor
QString inactiveColor
QList<std::shared_ptr<sh::ui::qt::QtFileView>> _views
QList<QWidget*> _mainviews
QHash<QWidget*, std::shared_ptr<QWidget>> _customwidgets
QList<sh::ui::qt::QtFileViewControl*> _viewcontrols
bool _skip_eventfilter = false

```

## Friends

```
friend class ui::qt::QtMainWindow
friend class ui::FileView

class QtFileView: public std::enable_shared_from_this<QtFileView>
    #include <qtfileview.h> Abstract base class for views for the contents of one directory.
    Subclassed by sh::ui::qt::QtFileIconview, sh::ui::qt::QtFileList
```

## Public Functions

```
QtFileView()
    Constructed only by the infrastructure and made available otherwise.

~QtFileView()

void setBackgroundColor (QString c)

QString backgroundColor ()

QList<std::shared_ptr<sh::filesystem::FilesystemNode>> selectedNodes ()

void setSelection (const      QList<std::shared_ptr<sh::filesystem::FilesystemNode>>
                    nodes)

std::shared_ptr<sh::filesystem::FilesystemNode> node ()

QList<std::shared_ptr<sh::filesystem::FilesystemNode>> getAllVisibleNodes ()

void setSizeFormatting (sh::filesystem::SizeFormatting v)

void gotoDir (std::shared_ptr<sh::filesystem::FilesystemNode> n)

void setHiddenFilesVisible (bool value)

QThread *thread ()

void configure (sh::ui::qt::QtFileViewControl *viewctrl)

void focus ()

void setSort (int column, Qt::SortOrder order)

QObject *as_qobject ()

QWidget *as_qwidget ()

QAbstractItemView *as_qabstractitemview ()

sh::ui::qt::QtFileViewControl *control ()
```

## Private Members

```
sh::ui::qt::QtFileViewControl *_viewctrl = nullptr

QString _tmp_bgColor

QPoint _dragStartPosition

int _async_gotodir_index = 0

bool _dragIsValid = false

int _sort_column = 0
```

```
Qt::SortOrder _sort_order = Qt::AscendingOrder
```

## Friends

```
friend class sh::actions::common::ActionHistoryNavigateForward
friend class sh::actions::common::ActionHistoryNavigateBackward
friend class sh::actions::common::ActionNavigateInHistory
friend class sh::ui::qt::QtFileViewControl
class EventFilter : public QObject
```

## Public Functions

```
EventFilter (QObject *parent, QtFileView *fileview, sh::ui::qt::QtFileViewControl *file-
viewctrl)
bool eventFilter (QObject *object, QEvent *event)
```

## Private Members

```
QtFileView *fileview
sh::ui::qt::QtFileViewControl *fileviewctrl
class QtFileViewControl : public sh::ui::FileView
#include <qtfileviewcontrol.h> Qt based file view.
```

## Public Functions

```
QtFileViewControl (sh::ui::qt::QtFilesystemPanel *panel, int i)
~QtFileViewControl ()
sh::ui::ColumnDimensions *columndimensions () override
Returns the column dimensions handler. .
sh::tools::HistoryTracker<std::shared_ptr<const sh::filesystem::Eurl>> *historyTracker ()
override
Returns the history tracker. .
FileViewMode viewmode () override
Returns the view mode (icons, list, ... ?). .
void setViewmode (FileViewMode m) override
Sets the view mode. .
int index () override
Returns the position of this file view within the panel. .
void gotoDir (std::shared_ptr<sh::filesystem::FilesystemNode> n) override
Jumps to a new current directory.
Implementations have to call the base class implementation inside.
sh::filesystem::SizeFormatting getSizeFormattingMode () override
Returns the mode how file sizes are formatted for displaying. .
```

```
void setSizeFormattingMode (sh::filesystem::SizeFormatting mode) override
    Sets the mode how file sizes are formatted for displaying. .

bool hiddenFilesVisible () override
    Returns if hidden files are visible. .

void setHiddenFilesVisible (bool v) override
    Sets the visibility of hidden files. .

int iconDimension () override
    Returns the icon size (in pixels). .

void setIconDimension (int v) override
    Sets the icon size (in pixels). .

void setSort (int column, Qt::SortOrder order) override
    Sets how to sort this view. .

int sortColumn () override
    Returns the index of the current sort column. .

Qt::SortOrder sortOrder () override
    Returns the current sort order. .

QList<std::shared_ptr<sh::filesystem::FilesystemNode>> selectedNodes () override
    Returns the nodes which the user has selected in this fileview. .

void setSelection (const      QList<std::shared_ptr<sh::filesystem::FilesystemNode>>
                    nodes) override
    Sets the node selection of this fileview. .

QList<std::shared_ptr<sh::filesystem::FilesystemNode>> getAllVisibleNodes ()
                                                                    override
    Returns a list of all nodes currently listed in this file view. .

void emit_nodesActivated (QList<std::shared_ptr<sh::filesystem::FilesystemNode>>
                        nodes)

void emit_selectionChanged ()

std::shared_ptr<sh::filesystem::FilesystemNode> node ()
    Returns the current directory.

void reload (bool skipModel = false)
    Reloads the data inside this file view.

sh::filesystem::FilesystemModelFileviewProxy *filemodel ()
    Returns the filesystem model for this view. .

void setFilemodel (sh::filesystem::FilesystemModelFileviewProxy *model)
    Sets the filesystem model for this view. .
```

## Signals

```
void nodesActivated (QList<std::shared_ptr<sh::filesystem::FilesystemNode>> nodes)

void selectionChanged ()
    Triggered when the selection have changed.

void viewOptionChanged ()
    Triggered when some view options have changed.
```



## Private Functions

void **setIndex** (int *i*)

## Private Members

*sh::ui::qt::QtFilesystemPanel* \*\_panel

int **i**

*sh::ui::ColumnDimensions* \*\_columndimensions

*sh::tools::HistoryTracker*<std::shared\_ptr<const *sh::filesystem::Eurl*>> \*\_historyTracker

*FileViewMode* **\_viewmode** = *FileViewMode::ListView*

bool **\_hiddenfilesvisible** = false

int **\_icondimension** = 0

int **\_sortColumn**

Qt::SortOrder **\_sortOrder**

*sh::filesystem::SizeFormatting* **\_sizeformattingmode** = *sh::filesystem::SizeFormatting::SizeFormattingModePrefix*

## Friends

**friend class** *sh::ui::qt::QtFilesystemPanel*

**class QtJumpBar** : public QWidget

*#include <qtjumpbar.h>* Button bar for navigating to places with parent-buttons and an text entry.

## Public Functions

**QtJumpBar** (QWidget \*parent = 0)

**~QtJumpBar** ()

void **setButtonMode** ()

void **setTextMode** ()

bool **isTextMode** ()

std::shared\_ptr<*sh::filesystem::FilesystemNode*> **node** ()

void **setNode** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> node)

QSize **sizeHint** () const

### Signals

```
void nodeChanged ()  
void eurlRequested (std::shared_ptr<const sh::filesystem::Eurl> eurl)
```

### Private Functions

```
void _buildButtons ()
```

### Private Members

```
std::shared_ptr<sh::filesystem::FilesystemNode> _node = 0  
Ui::QtJumpBar *ui  
bool _isTextMode
```

### Private Slots

```
void on_btnOk_clicked ()  
void on_btnAbort_clicked ()
```

```
class QtLinkButton : public QToolButton  
#include <qtlinkbutton.h> QToolButton with different look.  
Subclassed by sh::ui::qt::QtSearchPanelButton
```

### Public Functions

```
QtLinkButton (QWidget *parent = 0)  
void setSizeByFactor (int f)  
void setMenu (QStringList items, QString menuchangetxt = tr("(change)"))  
void setSelectedMenuItem (int idx)
```

### Signals

```
void menuItemSelected (int idx)
```

### Private Members

```
QString _linkcolor  
QStringList _menuitems  
QString _menuchangetxt
```

```
class QtLogViewDialog : public sh::ui::qt::QtDialog, public sh::ui::LogViewDialog  
#include <qtlogviewdialog.h> Qt based log view dialog.
```

## Public Functions

**QtLogViewDialog** (QString *headtext*, QString *subheadtxt*, *base::LogSeverity* *minseverity*)

**~QtLogViewDialog** ()

QString **headtext** ()

Returns the header text.

QString **subheadtxt** ()

Returns the 2nd header text.

*sh::base::LogSeverity* **minimumSeverity** ()

Returns the minimum severity.

qint64 **dialogId** ()

Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

bool **isInited** ()

Returns if this dialog is initialized.

Must be called in main thread.

bool **wasAccepted** ()

Returns if this dialog was 'accepted' by the user (typically by clicking some 'OK' button).

Must be called in main thread.

void **waitClosed** ()

Wait until the user closed the dialog in some way.

May be called in any thread.

void **close** ()

Closes the dialog.

Must be called in main thread.

bool **wasClosed** ()

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \***manager** ()

Returns the *DialogManager* which hosts this dialog.

## Private Functions

void **\_reloadLog** ()

void **setMinimumSeverity** (*sh::base::LogSeverity* *minseverity*)

## Private Members

*Ui::QtLogViewDialog* \*ui

## Private Slots

void on\_btnClose\_clicked()

void on\_btnSaveToFile\_clicked()

void on\_btnSeverity\_clicked()

**class** **QtMainWindow**: public QMainWindow, public *sh::ui::MainWindow*  
#include <qtmainwindow.h> The qt main window implementation.

## Public Functions

**QtMainWindow**(QWidget \*parent = 0)

**~QtMainWindow**()

void **initialize**() **override**

Initializes this main window. .

std::shared\_ptr<*DialogManager*> **dialogManager**() **override**

Returns the *DialogManager* which creates and drives *Dialogs* for this main window.

You typically should not need to use it directly.

int **fileViewCount**() **override**

Returns the current number of file views. .

void **addFileView**(bool *reinitialize* = false) **override**

Adds a new file view. .

void **removeFileView**(int *i*) **override**

Removes a file view. .

*sh::ui::FileView* \***currentFileView**() **override**

Returns the file view which is currently active (i.e. focussed). .

*sh::ui::FileView* \***getFileView**(int *i*) **override**

Returns a file view by position index. .

void **jumpToNode**(std::shared\_ptr<*sh::filesystem::FilesystemNode*> *node*) **override**

Let the current view jump to another location. .

void **setTitlePattern**(QString *value*) **override**

Sets the window title pattern. .

void **setTreeVisible**(bool *v*) **override**

Sets the visibility of the directory tree. .

void **setFileDetailsPanelVisible**(bool *v*) **override**

Sets the visibility of the file details panel. .

std::shared\_ptr<*sh::filesystem::FilesystemNode*> **currentDirectory**() **override**

Gets the *sh::filesystem::FilesystemNode* selected in the current view. .

```

std::shared_ptr<sh::ui::ActionExecutionInfoPanel> addInfoPanel (int position,
                                                             sh::actions::ActionExecutionInfo
                                                             *info, QColor color =
                                                             QColor()) override

    Creates a new action execution panel.

    Let this smart pointer die for removing it.

void jumpbarSetTextMode () override
    Sets the jumpbar to text input mode. .

void jumpbarSetButtonMode () override
    Sets the jumpbar to button mode. .

bool jumpbarIsTextMode () override
    Returns if the jumpbar is in text input mode. .

QString toolbarPosition () override
    Returns the current toolbar position. .

void setToolbarPosition (QString pos) override
    Sets the toolbar position. .

QString detailPanelPosition () override
    Returns the current detail panel position. .

void setDetailPanelPosition (QString pos) override
    Sets the detail panel position. .

std::shared_ptr<AboutDialog> showAboutDialog () override
    Shows and returns an 'About' dialog. .

std::shared_ptr<ManageProfilesDialog> showManageProfilesDialog () override
    Shows and returns a 'Manage Saved Settings' dialog. .

std::shared_ptr<OpenWithDialog> showOpenWithDialog (std::shared_ptr<sh::filesystem::FilesystemNode>
                                                    node,
                                                    QList<std::shared_ptr<sh::tools::filetypes::OpenMethod>
                                                    openMethods) override

    Shows and returns a 'Open With' dialog. .

std::shared_ptr<StoreProfileDialog> showStoreProfileDialog (std::shared_ptr<const
                                                           sh::filesystem::Eurl>
                                                           eurl) override

    Shows and returns a 'Save Settings' dialog. .

std::shared_ptr<TuningDialog> showTuningDialog () override
    Shows and returns a 'Tuning' dialog. .

std::shared_ptr<LogViewDialog> showLogViewDialog (QString headtext = QString(),
                                                  QString subheadtxt = QString(),
                                                  sh::base::LogSeverity minseverity =
                                                  sh::base::LogSeverity::_DEFAULT)
                                                  override

    Shows and returns a 'Log' dialog. .

std::shared_ptr<ManageBookmarksDialog> showManageBookmarksDialog ()
                                                    override

    Shows and returns a 'Manage Bookmarks' dialog. .

std::shared_ptr<FilePropertyDialog> showFilePropertyDialog (QList<std::shared_ptr<sh::filesystem::Filesystem
                                                            nodes) override

    Shows and returns a 'File Properties' dialog. .

```

`std::shared_ptr<ActionExecutionInfoDialog> createActionExecutionInfoDialog (sh::actions::ActionExecutionInfoDialog *info)` **override**  
Creates an action execution dialog. .

void **handleCloseAppRejected** (QString *rejectmsg*) **override**  
React on a rejected application close request (with some feedback message). .

void **handleClosed** () **override**  
Do some cleanup steps just when closing starts. Implementations must call base implementation!

sh::ui::qt::QtUIStyle \***uiStyle** ()  
Gets a *QtUIStyle* instance, which helps for common ui styling.

void **\_initialize** (sh::base::SingletonInitializer \*singletonInitializer)  
Initializes the main window. For custom initialization logic, see MainWindow.initialize. .

void **fileViewsReloadAll** ()  
Reload all content in each file view.

void **onFileViewOptionsChanged** (std::function<void> int  
> *fcn*, QObject \**owner* = 0) Sets a handler for some view options in a file view changed (optionally bound to an owner lifetime).

void **onCurrentFileViewChanged** (std::function<void>  
> *fcn*, QObject \**owner* = 0) Sets a handler for the currently active file view changed (optionally bound to an owner lifetime).

void **onFileViewCountChanged** (std::function<void>  
> *fcn*, QObject \**owner* = 0) Sets a handler for the number of file views changed (optionally bound to an owner lifetime).

void **onCurrentDirectoryChanged** (std::function<void>  
> *fcn*, QObject \**owner* = 0) Sets a handler for the current directory in the active file view changed (optionally bound to an owner lifetime).

void **onGlobalViewOptionsChanged** (std::function<void>  
> *fcn*, QObject \**owner* = 0) Sets a handler for some global view options changed (optionally bound to an owner lifetime).

void **onCurrentProfileChanged** (std::function<void>  
> *fcn*, QObject \**owner* = 0) Sets a handler for the current profile changed (optionally bound to an owner lifetime).

void **jumpToEurl** (std::shared\_ptr<const sh::filesystem::Eurl> *eurl*)  
Let the current view jump to another location. .

QString **titlePattern** ()  
Returns the window title pattern.

void **setCurrentProfile** (QString *profile*)  
Sets the current profile. .

QString **currentProfile** ()  
Returns the current profile.

void **reloadProfile** ()  
Reloads the profile data.

bool **treeVisible** ()  
Returns the visibility of the directory tree.

void **setTreeSticky** (bool *v*)  
 Sets if the directory tree should follow the active file view. .

bool **treeSticky** ()  
 Returns if the directory tree follows the active file view.

bool **fileDetailsPanelVisible** ()  
 Returns the visibility of the file details panel.

std::shared\_ptr<*sh::ui::ActionExecutionInfoPanel*> **addInfoPanel** (*sh::actions::ActionExecutionInfo* \**info*, QColor *color* = QColor())  
 Creates a new action execution panel.  
 Let this smart pointer die for removing it.

std::shared\_ptr<*ActionExecutionInfoPanel*> **showErrorPanel** ()  
 Shows an error panel.

void **\_closeDirectly** ()  
 Directly begin application shutdown without checks.

bool **closeApp** ()  
 Check if the application may shut down now, and if so, initiate it.

bool **isCloseable** (QString \**msg* = nullptr)  
 Check if the application may shut down now.

bool **isClosing** ()  
 Returns if the application is currently closing.

## Public Static Functions

bool **tryCreateMainWindow** (*MainWindow* \*&*mainWindow*)  
*QtMainWindow* \***mainWindow** ()

void **setMainWindow** (*MainWindow* \**mainWindow*)  
 Sets the main window instance. .

bool **isReady** ()  
 Returns if this process ui is ready (i.e. is created or runs headless).

bool **runsHeadless** ()  
 Returns if this process runs headless, i.e. without any ui, just for a background process.

void **\_closeDirectly** (*sh::base::SingletonInitializer* \**singletonInitializer*)

QString **uiMode** ()  
 Returns the UI mode (e.g. qt, web).

### Private Functions

```
void _setupGlobalShortcut (std::shared_ptr<sh::actions::AbstractActionItem> action)
void _setupGlobalShortcut_recursive (std::shared_ptr<sh::actions::SubmenuActionItem>
                                     submenu)
void _addPermanentActionToToolbar (std::shared_ptr<sh::actions::SubmenuActionItem>
                                   a, bool rightSide = false, bool preventDefault-
                                   Action = false)
void _refreshToolbar (std::shared_ptr<sh::actions::ActionInstantiation>)
void _jumpToNode (std::shared_ptr<sh::filesystem::FilesystemNode> node, int skip)
void _jumpToIndex (QModelIndex idx, int skip)
void _newToStatusbar_helper (sh::ui::qt::QtActionExecutionInfoPanel *w)
void _refresh_detailthumbnailvisibility ()
void _thumbnail_resized ()
void _resizethumbnail ()
void _detailpanel_resized ()
void _resizedetailpanel ()
void requestDetailThumbnail (bool force = true)
void setStatusbarVisibility (bool v)
```

### Private Members

```
Ui::QtMainWindow *ui
sh::ui::qt::QtFilesystemPanel *fspanel
sh::filesystem::FilesystemModelDirectoryTreeProxy *treemodel = 0
sh::ui::qt::QtJumpBar *jumpbar
QTimer _statusbar_timer
int _statusbar_fullheight
int _statusbar_targetheight = 0
QSet<sh::ui::qt::QtActionExecutionInfoPanel*> _statusbar_childs
std::shared_ptr<sh::filesystem::FilesystemNode> _currentDirectory = 0
bool _request_detailthumbnail_skip = false
int _thumbnailwidth = 0
Qt::Orientation _myorientation = Qt::Horizontal
sh::ui::qt::QtUIStyle *_uistyle = 0
QAction *_toolbarLeftRightSplitSeparator
QHash<QString, std::shared_ptr<sh::actions::common::ActionGroups>> actionGroupsActions
QList<QtToolBarButtonHandler*> toolbarButtonHandlers
std::shared_ptr<QtDialogManager> _dialogManager
```



### Private Slots

```
void slot_detailbar_moved ()
void slot_toolbar_moved ()
void slot_statusbartimer ()
void slot_treeview_collapsedexpanded (const QModelIndex &index)
```

### Private Static Attributes

```
QtMainWindow *_qtMainWindow = nullptr
```

### Friends

```
friend class QtToolBarButton
friend class QtFilesystemPanel
class MyFlexibleLabel : public QLabel
    #include <qtmainwindow.h> Needed internally in main window in order to have a label which
    really does not request any sizes.
```

### Public Functions

```
MyFlexibleLabel (QWidget *parent = 0)
QSize sizeHint () const override
class QtManageBookmarksDialog : public sh::ui::qt::QtDialog, public sh::ui::ManageBookmarksDialog
    #include <qtmanagebookmarksdialog.h> Qt based manage bookmarks dialog.
```

### Public Functions

```
QtManageBookmarksDialog ()
~QtManageBookmarksDialog ()
qint64 dialogId ()
    Returns the dialog id.
    Each instance has an id unique in the complete Shallot process lifetime.
    Must be called in main thread.
bool isInitiated ()
    Returns if this dialog is initialized.
    Must be called in main thread.
bool wasAccepted ()
    Returns if this dialog was ‘accepted’ by the user (typically by clicking some ‘OK’ button).
    Must be called in main thread.
void waitClosed ()
    Wait until the user closed the dialog in some way.
    May be called in any thread.
```

void **close** ()

Closes the dialog.

Must be called in main thread.

bool **wasClosed** ()

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \***manager** ()

Returns the *DialogManager* which hosts this dialog.

### Private Functions

void **readBookmarks** ()

void **\_refresh\_values** ()

void **\_selectbookmark** (QString *id*)

void **\_selectfolder** (QStringList *name*)

### Private Members

*Ui::QtManageBookmarksDialog* \***ui**

### Private Slots

void **on\_btnClose\_clicked** ()

void **on\_treeWidget\_itemSelectionChanged** ()

void **on\_btnDelete\_clicked** ()

void **on\_edtLabel\_textChanged** (const QString &*arg1*)

void **on\_edtLocation\_textChanged** (const QString &*arg1*)

void **on\_btnStore\_clicked** ()

void **on\_btnNew\_clicked** ()

void **on\_btnUp\_clicked** ()

void **on\_btnDown\_clicked** ()

void **on\_btnMove\_clicked** ()

**class** QtManageProfilesDialog : **public** *sh::ui::qt::QtDialog*, **public** *sh::ui::ManageProfilesDialog*  
#include <qtmanageprofilesdialog.h> Qt based manage saved settings dialog.

## Public Functions

**QtManageProfilesDialog()**

**~QtManageProfilesDialog()**

**qint64 dialogId()**

Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

**bool isInitiated()**

Returns if this dialog is initialized.

Must be called in main thread.

**bool wasAccepted()**

Returns if this dialog was 'accepted' by the user (typically by clicking some 'OK' button).

Must be called in main thread.

**void waitClosed()**

Wait until the user closed the dialog in some way.

May be called in any thread.

**void close()**

Closes the dialog.

Must be called in main thread.

**bool wasClosed()**

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \***manager()**

Returns the *DialogManager* which hosts this dialog.

## Private Functions

**void \_createProfileList()**

**void \_createSettingsList()**

**void \_createNodesList()**

## Private Members

*Ui::*QtManageProfilesDialog \***ui**

QStringListModel \***\_model**

*sh::settings::ProfileNode* \***\_currentNode = 0**

QHash<QListWidgetItem\*, QString> **\_nodes**

### Private Slots

```
void on_comboBox_currentTextChanged (const QString &arg1)
void on_btnClose_clicked ()
void on_btnDelNode_clicked ()
void on_btnDelProfile_clicked ()
void slot_listViewactivated (QListWidgetItem*, QListWidgetItem*)
```

```
class QtOpenWithDialog : public sh::ui::qt::QtDialog, public sh::ui::OpenWithDialog
#include <qtopenwithdialog.h> Qt based open with dialog.
```

### Public Functions

```
QtOpenWithDialog (std::shared_ptr<sh::filesystem::FilesystemNode> node,
                  QList<std::shared_ptr<sh::tools::filetypes::OpenMethod>> open-
                  Methods)
~QtOpenWithDialog ()
void setProgramList (QList<std::shared_ptr<sh::tools::filetypes::OpenMethod>> open-
                  Methods)
std::shared_ptr<sh::tools::filetypes::OpenMethod> chosenMethod () override
    Returns the chosen open-method (program for opening the file).
bool rememberForMimetype () override
    Returns if the chosen method is checked to be remembered for the same mime-type in the future.
std::shared_ptr<const sh::filesystem::Eurl> rememberForDirectory () override
    If it's checked for the chosen method to be remembered for some subdirectory in the future, it
    returns the Eurl of this subdirectory, otherwise nullptr.
bool rememberForFile () override
    Returns if the chosen method is checked to be remembered for the same file in the future.
std::shared_ptr<sh::filesystem::FilesystemNode> node ()
    Returns the file node which is later to be opened.
QList<std::shared_ptr<sh::tools::filetypes::OpenMethod>> openMethods ()
    Returns the open-methods (programs for opening the file) to present for choice.
qint64 dialogId ()
    Returns the dialog id.
    Each instance has an id unique in the complete Shallot process lifetime.
    Must be called in main thread.
bool isInitiated ()
    Returns if this dialog is initialized.
    Must be called in main thread.
bool wasAccepted ()
    Returns if this dialog was 'accepted' by the user (typically by clicking some 'OK' button).
    Must be called in main thread.
```

void **waitClosed** ()  
 Wait until the user closed the dialog in some way.  
 May be called in any thread.

void **close** ()  
 Closes the dialog.  
 Must be called in main thread.

bool **wasClosed** ()  
 Returns if this dialog was closed.  
 Must be called in main thread.

*DialogManager* \***manager** ()  
 Returns the *DialogManager* which hosts this dialog.

### Private Members

std::shared\_ptr<*sh::tools::filetypes::OpenMethod*> **\_chosenMethod**  
 std::shared\_ptr<const *sh::filesystem::Eurl*> **\_rememberfordirectory**  
 QAbstractItemModel \* **\_listmodel**  
*Ui::QtOpenWithDialog* \***ui**

### Private Slots

void **on\_btnCancel\_clicked** ()  
 void **on\_btnOk\_clicked** ()  
 void **on\_btnSelectOther\_clicked** ()  
 void **on\_listView\_activated** (const QModelIndex &*index*)  
 void **on\_btnAnotherAncestor\_clicked** ()  
 void **on\_chkRememberDir\_toggled** (bool *checked*)  
**class QtOpenWithDialogModel** : public QStringListModel  
*#include <qopenwithdialog.h>* Used internally in OpenWithDialogModel.

### Public Functions

**QtOpenWithDialogModel** (QList<std::shared\_ptr<*sh::tools::filetypes::OpenMethod*>>  
                           *methods*, QObject \**parent* = 0)  
 QVariant **data** (const QModelIndex &*index*, int *role*) const

### Private Members

QList<std::shared\_ptr<*sh::tools::filetypes::OpenMethod*>> **\_methods**

**class QtSearchPanel** : public QWidget  
#include <qtsearchpanel.h> Search panel for usage in the Qt main window.

### Public Functions

**QtSearchPanel** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> node, QWidget \*parent = 0)

**~QtSearchPanel** ()

### Friends

**friend class** QtSearchPanelConfiguration

template<class **T1**, class **T2**>

**class QtSearchPanelAbstractEditor** : public *T1*, public *T2*  
#include <qtsearchpanelconfiguration.h> Helper for other Qt based search panel editors.

### Public Functions

**QtSearchPanelAbstractEditor** (QWidget \*parent = 0)

bool **isEnabled** ()

void **setEnabled** (bool v)

**class QtSearchPanelButton** : public *sh::ui::qt::QtLinkButton*, public *sh::ui::SearchPanelButton*  
#include <qtsearchpanelconfiguration.h> Qt based *SearchPanelButton*.

### Public Functions

**QtSearchPanelButton** (QWidget \*parent = 0)

void **setButtonText** (QString txt)  
Sets the button text. .

void **setMenuSelection** (int i)  
Sets the currently selected menu item (for buttons with menus). .

void **setSizeByFactor** (int f)

void **setMenu** (QStringList items, QString menuchangetxt = tr("(change)"))

void **setSelectedItem** (int idx)

## Signals

void **menuItemSelected** (int *idx*)

**class** **QtSearchPanelConfiguration**: public *sh::ui::SearchPanelConfiguration*  
*#include <qtsearchpanelconfiguration.h>* Search panel configuration for usage in a Qt ui.

## Public Functions

**QtSearchPanelConfiguration** (*QtSearchPanel* \**owner*, QHBoxLayout \**editors*)

*SearchPanelButton* \***addMenuButton** (QString *text*, QStringList *menu*, std::function<void> int  
 > *onchanged*) Adds and returns a button for a menu. .

*SearchPanelButton* \***addActionButton** (QString *text*, std::function<void>  
 > *action*) Adds and returns a button for an action. .

*SearchPanelTextEditor* \***addTextEditor** ()  
 Adds and returns a text editor. .

*SearchPanelDateTimeEditor* \***addDateTimeEditor** ()  
 Adds and returns a date/time editor. .

*SearchPanelLabelEditor* \***addLabel** ()  
 Adds and returns a label. .

*SearchPanelSpacerEditor* \***addSpacer** ()  
 Adds and returns a spacer. .

*SearchPanelAbstractEditor* \***getEditorAt** (int *i*)  
 Returns the editor widget at a given position. .

void **onDestroyed** (std::function<void>  
 > *fcn*) *QObject* \**owner* = 0 Sets a handler for panel removal (optionally bound to an owner lifetime).

## Public Members

*\_SearchPanelConfiguration\_HelperQObject* **myqobject**

## Private Members

*QtSearchPanel* \***owner**

QHBoxLayout \***editors**

**class** **QtSearchPanelDateTimeEditor**: public *sh::ui::qt::QtSearchPanelAbstractEditor*<QDateTimeEdit, *SearchPanelDateTimeEditor*>  
*#include <qtsearchpanelconfiguration.h>* Qt based *SearchPanelDateTimeEditor*.

### Public Functions

```
QtSearchPanelDateTimeEditor (QWidget *parent = 0)

QDateTime datetime ()
    Returns the selected date/time. .

void setDatetime (QDateTime s)
    Sets the selected date/time.

bool isWidgetEnabled ()

void setWidgetEnabled (bool v)

class QtSearchPanelLabelEditor : public sh::ui::qt::QtSearchPanelAbstractEditor<QLabel, sh::ui::SearchPanelLabelEditor>
    #include <qtsearchpanelconfiguration.h> Qt based SearchPanelLabelEditor.
```

### Public Functions

```
QtSearchPanelLabelEditor (QWidget *parent = 0)

QString textContent ()
    Returns the text content. .

void setTextContent (QString s)
    Sets the text content. .

SearchPanelAbstractEditor *focusProxyEditor ()
    Returns the focus proxy widget (which focus gets redirected to in some situations). .

void setFocusProxyEditor (SearchPanelAbstractEditor *w)
    Sets the focus proxy widget (which focus gets redirected to in some situations). .

bool isWidgetEnabled ()

void setWidgetEnabled (bool v)

class QtSearchPanelSpacerEditor : public sh::ui::qt::QtSearchPanelAbstractEditor<QLabel, sh::ui::SearchPanelSpacerEditor>
    #include <qtsearchpanelconfiguration.h> Qt based SearchPanelSpacerEditor.
```

### Public Functions

```
QtSearchPanelSpacerEditor (QWidget *parent = 0)

bool isWidgetEnabled ()

void setWidgetEnabled (bool v)

class QtSearchPanelTextEditor : public sh::ui::qt::QtSearchPanelAbstractEditor<QLineEdit, sh::ui::SearchPanelTextEditor>
    #include <qtsearchpanelconfiguration.h> Qt based SearchPanelTextEditor.
```



## Public Functions

```

QtSearchPanelTextEditor (QWidget *parent = 0)

QString textContent ()
    Returns the text content. .

void setTextContent (QString s)
    Sets the text content. .

QString placeholderDescription ()
    Returns the placeholder description text (visible if field is empty). .

void setPlaceholderDescription (QString s)
    Sets the placeholder description text (visible if field is empty). .

bool isWidgetEnabled ()

void setWidgetEnabled (bool v)

class QtSettingUIFrame : public QFrame
    #include <qtsettinguiframe.h> User interface for one handling one setting.

```

## Public Functions

```

QtSettingUIFrame (sh::settings::Setting *setting, bool withcheckbox, QString displayvalue,
                  QString additionalCheck, QVariant additionalCheckValue, QWidget
                  *parent = 0)

bool isChecked ()

void setChecked (bool val)

void setAdditionalCheckVisible (bool v)

QVariant additionalCheckValue ()

```

## Private Members

```

QCheckBox *_checkbox = 0

QCheckBox *_additionalcheckbox = 0

QLabel *_additionalcheckboxlabel = 0

QWidget *_additionalcheckwidget = 0

QVariant _additionalCheckValue

class QtStoreProfileDialog : public sh::ui::qt::QtDialog, public sh::ui::StoreProfileDialog
    #include <qtstoreprofiledialog.h> Qt based save settings dialog.

```

## Public Functions

**QtStoreProfileDialog** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*)

**~QtStoreProfileDialog** ()

**QList<*sh::ui::StoreProfileDialog::SettingEntry*> checkedSettings** () **override**

Returns the list of settings the user has checked to store.

**QString profileName** () **override**

Returns the profile the user has chosen to store settings for.

**bool withSubDirectories** () **override**

Returns if the user has chosen to apply the checked settings also for subdirectories.

**QStringList inheritsFrom** () **override**

Returns the list of profiles the user has chosen to inherit from.

**bool hasGlobal** () **override**

Returns if the user has chosen to apply the checked settings for each directory (instead of the current directory).

**std::shared\_ptr<const *sh::filesystem::Eurl*> eurl** ()

Returns the current directory this dialog shall work on.

**qint64 dialogId** ()

Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

**bool isInitd** ()

Returns if this dialog is initialized.

Must be called in main thread.

**bool wasAccepted** ()

Returns if this dialog was 'accepted' by the user (typically by clicking some 'OK' button).

Must be called in main thread.

**void waitClosed** ()

Wait until the user closed the dialog in some way.

May be called in any thread.

**void close** ()

Closes the dialog.

Must be called in main thread.

**bool wasClosed** ()

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \***manager** ()

Returns the *DialogManager* which hosts this dialog.

### Private Functions

```
void _settingsLevel (int level, bool persist = true)
void _setglobal (bool v)
```

### Private Members

```
QHash<sh::ui::qt::QtSettingUIFrame*, std::shared_ptr<sh::settings::Setting>> _widgets
QHash<int, QLabel*> _headerlabels
QStringList _inheritProfileNameList
bool _global = false
Ui::QtStoreProfileDialog *ui
int _level
```

### Private Slots

```
void on_btnAddProf_clicked ()
void on_btnGlobal_clicked ()
void on_btn22_clicked ()
void on_toolButton_2_clicked ()
void on_toolButton_clicked ()
```

```
class QtToolBarButton : public QToolButton
    #include <qttoolbarbutton.h> A qt toolbar button implementation with optional submenu.
```

### Public Types

```
enum Position
    Values:
        enumerator LEFT
        enumerator RIGHT
        enumerator TOP
        enumerator BOTTOM
```

### Public Functions

```
QtToolBarButton (QWidget *parent = 0)
void setText (const QString &text)
QString text () const
void setIcon (const QIcon &icon)
QIcon icon ()
QSize sizeHint () const
```

```
void setButtonText (QString v)
QString buttonText ()
void setButtonIcon (QIcon v)
QIcon buttonIcon ()
void setButtonEnabled (bool v)
bool isButtonEnabled ()
bool hasExpander ()
void setSubmenu (QtActionMenu *menu)
QtActionMenu *submenu ()
void setLocation (Position location)
void setClickAction (std::function<void>
    > action)
void unsetClickAction ()
void trigger ()
void openMenu ()
```

### Private Functions

```
void _calcDims ()
void computeArrowAndDividerLineCoordinates ()
```

### Private Members

```
QString _text
QString _text1
QString _textWithoutMnemonic1
QString _text2
QString _textWithoutMnemonic2
QIcon _icon
bool _drawIconOnly = false
bool _hovered = false
bool _arrowhovered = false
sh::ui::qt::QtActionMenu *_menu = 0
std::function<void ()> _clickAction
Position location
int tw1
int tworig1
int tw2
```

```
int tworig2
int twmax
int tworigmax
int th
int thorig
int iw
int ih
int iedim
int ew
int eh
const int textpad = 20
const int opad = 3
int expx1
int expx2
int expy1
int expy2
int xt
int yt1
int yt2
int xi
int yi
QPainterPath arrow
bool _drawmnemonics = false
```

### Private Slots

```
void slot_clicked()
```

### Friends

```
friend class sh::actions::ActionsManager
```

```
class QtToolbarButtonHandler
```

*#include <qttoolbarbuttonhandler.h>* A handler which reflects the *sh::actions* objects to a graphical toolbar button (and keeps that up-to-date).

### Public Functions

```
QtToolBarButtonHandler (std::shared_ptr<sh::actions::SubmenuItem> action,  
                        bool preventDefaultAction, QtToolBarButton *button, QAc-  
                        tion *qaction)  
  
~QtToolBarButtonHandler ()
```

### Private Functions

```
void __updateSubmenu ()
```

### Private Members

```
std::shared_ptr<sh::actions::SubmenuItem> action  
  
QtActionMenu menu  
  
bool preventDefaultAction  
  
QtToolBarButton *button
```

### Private Static Functions

```
void __applyPropertiesToButton (sh::actions::SubmenuItem *action, QtToolBar-  
                             Button *button, QAction *qaction)
```

```
class QtTuningDialog: public sh::ui::qt::QtDialog, public sh::ui::TuningDialog  
    #include <qttuningdialog.h> Qt based tuning dialog.
```

### Public Functions

```
QtTuningDialog ()  
  
~QtTuningDialog ()  
  
qint64 dialogId ()  
    Returns the dialog id.  
  
    Each instance has an id unique in the complete Shallot process lifetime.  
  
    Must be called in main thread.  
  
bool isInitiated ()  
    Returns if this dialog is initialized.  
  
    Must be called in main thread.  
  
bool wasAccepted ()  
    Returns if this dialog was 'accepted' by the user (typically by clicking some 'OK' button).  
  
    Must be called in main thread.  
  
void waitClosed ()  
    Wait until the user closed the dialog in some way.  
  
    May be called in any thread.
```

void **close** ()

Closes the dialog.

Must be called in main thread.

bool **wasClosed** ()

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \***manager** ()

Returns the *DialogManager* which hosts this dialog.

## Private Types

**typedef** QPair<QString, QWidget\*> **BoxWidgetByDesc**

## Private Functions

QString **\_changedlglabel** (std::shared\_ptr<*sh::configuration::ConfigurationValue*> cv)

void **do\_change** (std::shared\_ptr<*sh::configuration::ConfigurationValue*> cv)

void **filterVisibility** (QString s)

## Private Members

*Ui::QtTuningDialog* \***ui**

QList<*BoxWidgetByDesc*> **boxWidgetsByDesc**

## Private Slots

void **on\_btnCancel\_clicked** ()

void **on\_lineEditSearchInt\_textChanged** (const QString &arg1)

void **on\_lineEditSearchExt\_textChanged** (const QString &arg1)

void **on\_lineEditSearchBehav\_textChanged** (const QString &arg1)

void **on\_lineEditSearchAppear\_textChanged** (const QString &arg1)

void **on\_tabWidget\_currentChanged** (int index)

**class** QtUIStyle

*#include* <*qtuiestyle.h*> Methods for applying the Shallot visible style.

See *sh::ui::qt::QtMainWindow* about how to get an instance.

## Public Functions

**QtUIStyle** (*sh::ui::qt::QtMainWindow \*mainwnd*)

Constructed only by the infrastructure and made available otherwise.

**QColor windowColor** ()  
**QColor backgroundColor** ()  
**QColor inactiveBackgroundColor** ()  
**QColor windowColorHighlighted** ()  
**QColor windowColorHalfHighlighted** ()  
**QColor windowColorDarkened** ()  
**QColor windowColorLikeTitlebar** ()  
**QColor linkColor** ()  
**QColor titlebarColor** ()  
**QColor foregroundColor** ()  
**QColor foregroundColorLighter1** ()  
**QColor foregroundColorLighter2** ()  
**int fontSizeInPt** (int *r* = 100)  
**QColor mix** (float *n1*, **QColor** *c1*, **QColor** *c2*)  
*Sheet* **createSheet** ()

## Private Members

*sh::ui::qt::QtMainWindow* \*\_mainWindow

**class Sheet**  
    #include <qtuistyle.h>

## Public Functions

**QString sheet** ()  
*Sheet* **customcss** (QString *css*)  
*Sheet* **fontsize\_byFactor** (int *f*)  
*Sheet* **fontsize\_header** ()  
*Sheet* **fontsize\_smaller** ()  
*Sheet* **textcolor** (QColor *color*)  
*Sheet* **textcolor\_lighter1** ()  
*Sheet* **textcolor\_lighter2** ()  
*Sheet* **background** (QColor *color*, bool *restrictQWidget* = true)  
*Sheet* **background\_highlighted** (bool *restrictQWidget* = true)  
*Sheet* **background\_halfhighlighted** (bool *restrictQWidget* = true)



*Sheet* **background\_darkened** (bool *restrictQWidget* = true)  
*Sheet* **background\_liketitlebar** (bool *restrictQWidget* = true)  
*Sheet* **applyTo** (QWidget \**widget*)  
*Sheet* **bold** ()

### Private Functions

**Sheet** (*QtUIStyle* \**style*, QString *sheet*)

### Private Members

QString **\_sheet**  
*QtUIStyle* \***\_style**

### Friends

**friend class** QtUIStyle

### namespace feedbackpanels

Qt user interface parts for user feedback in action execution dialogs.

Implementations of *sh::ui::qt::feedbackpanels::FeedbackPanel* are used for implementing parts of *sh::ui::ActionExecutionInfoDialog*.

### Typedefs

**typedef** QPair<QString, QWidget\*> **GridFormRow**

**class Credentials** : **public** *sh::ui::qt::feedbackpanels::FeedbackPanel*  
*#include <credentials.h>* *FeedbackPanel* for user login credentials (password based).

### Public Functions

**Credentials** (QString *domain*, QString *username*, QString *password*, QString *text*, bool *showDomain*, bool *showUsername*, bool *showPassword*, bool *showAnonymous*, bool *showRemember*)  
**~Credentials** ()  
void **cancelRequested** ()  
bool **isClosed** ()  
void **waitUntilClosed** ()  
int **preferredHeight** ()

## Public Members

QString **domain**  
QString **username**  
QString **password**  
bool **anonymous**  
bool **remember**  
bool **accepted**

## Signals

void **wasClosed**()

## Private Members

*Ui::*Credentials \***ui**

## Private Slots

void **on\_chkAnonymous\_toggled**(bool *checked*)  
void **on\_pushButton\_3\_clicked**()  
void **on\_pushButton\_4\_clicked**()

**class FeedbackPanel** : public QWidget  
*#include <feedbackpanel.h>* Abstract base class for a Qt helper widget used in action user feedback.  
  
Subclassed by *sh::ui::qt::feedbackpanels::Credentials*, *sh::ui::qt::feedbackpanels::GridForm*, *sh::ui::qt::feedbackpanels::MsgBox*, *sh::ui::qt::feedbackpanels::UnixPermissions*

## Public Functions

**FeedbackPanel**(QWidget \**parent* = 0)  
bool **isClosed**()  
void **waitUntilClosed**()  
int **preferredHeight**()  
void **cancelRequested**() = 0

## Signals

void **wasClosed** ()

## Private Members

QMutex **\_mutex**

QWaitCondition **\_closedcondition**

bool **\_closed** = false

**class GridForm**: public *sh::ui::qt::feedbackpanels::FeedbackPanel*  
#include <gridform.h> *FeedbackPanel* for grid based forms.

## Public Functions

**GridForm**(QString *text*, QList<*sh::ui::qt::feedbackpanels::GridFormRow*> *form*,  
QStringList *answers*, int *defaultanswer* = -1, int *cancelanswer* = -1, QWidget  
\**parent* = 0)

**~GridForm** ()

int **preferredHeight** ()

void **cancelRequested** ()

bool **isClosed** ()

void **waitUntilClosed** ()

## Public Members

int **answer** = -1

## Signals

void **wasClosed** ()

## Private Members

*Ui::GridForm* \***ui**

QHash<QPushButton\*, int> **btn2index**

QPushButton \***\_cancelBtn** = 0

QPushButton \***\_defaultBtn** = 0

### Private Slots

```
void slot_btnclicked()
```

```
class GridFormInnerSimpleChooser : public QWidget
#include <gridforminnersimplechooser.h> Helper widget for a GridForm.
```

### Public Functions

```
GridFormInnerSimpleChooser (sh::actions::ActionExecutionUserFeedback::Choices
                             *choices, QWidget *parent = 0)
```

```
~GridFormInnerSimpleChooser ()
```

### Public Members

```
int answer = -1
```

```
QString text
```

### Private Functions

```
bool eventFilter (QObject *obj, QEvent *event)
```

### Private Members

```
QLineEdit *lineedit
```

```
sh::actions::ActionExecutionUserFeedback::Choices *choices
```

```
QVBoxLayout *mylayout
```

```
sh::actions::ActionExecutionUserFeedback::Choice *currentchoice = 0
```

### Private Slots

```
void slot_chosen (int id)
```

```
void slot_textedited (QString t)
```

```
class MsgBox : public sh::ui::qt::feedbackpanels::FeedbackPanel
#include <msgbox.h> FeedbackPanel for showing a message, some answer buttons, and option-
ally a text input box.
```

### Public Functions

```
MsgBox (QString question, QList<QString> answers, QString icon = "", QString with-
InputBox = QString(), bool inputBoxMultiline = false, int defaultanswer = -1,
int cancelanswer = -1, int valuePreselectFrom = -1, int valuePreselectTo = -1,
QList<QString> answericons = QList<QString>(), QWidget *parent = 0)
```

```
~MsgBox ()
```

```
void cancelRequested ()
```

```
bool isClosed ()
```

```
void waitUntilClosed ()
```

## Public Members

```
int answer = -1
```

```
QString text
```

## Signals

```
void wasClosed ()
```

## Private Members

```
Ui::MsgBox *ui
```

```
QHash<QPushButton*, int> btn2index
```

```
QWidget *_initialFocusWidget = 0
```

```
QPushButton *_cancelBtn = 0
```

```
QPushButton *_defaultBtn = 0
```

```
bool _inputBoxMultiline
```

```
int _valuePreselectFrom
```

```
int _valuePreselectTo
```

## Private Slots

```
void slot_btnclicked ()
```

```
class UnixPermissions : public sh::ui::qt::feedbackpanels::FeedbackPanel  
    #include <unixpermissions.h> FeedbackPanel for setting one set of unix filesystem permissions.
```

## Public Functions

```
UnixPermissions (QWidget *parent = 0)
```

```
~UnixPermissions ()
```

```
void cancelRequested ()
```

```
void setflags (bool userMayRead, bool userMayWrite, bool userMayExecute, bool group-  
    MayRead, bool groupMayWrite, bool groupMayExecute, bool others-  
    MayRead, bool othersMayWrite, bool othersMayExecute, bool sticky, bool  
    setuid, bool setgid)
```

```
void setusers (QStringList users, QString selecteduser)
```

```
void setgroups (QStringList groups, QString selectedgroup)
```

```
bool isClosed ()
```

```
void waitUntilClosed ()
```

```
int preferredHeight ()
```

### Public Members

bool **accepted**  
bool **userMayRead**  
bool **userMayWrite**  
bool **userMayExecute**  
bool **groupMayRead**  
bool **groupMayWrite**  
bool **groupMayExecute**  
bool **othersMayRead**  
bool **othersMayWrite**  
bool **othersMayExecute**  
bool **sticky**  
bool **setuid**  
bool **setgid**  
QString **user**  
QString **group**

### Signals

void **wasClosed**()

### Private Members

*Ui::*UnixPermissions \***ui**

### Private Slots

void **on\_pushButton\_clicked**()  
void **on\_pushButton\_2\_clicked**()

### namespace **web**

The web user interface implementation.

This optional UI provides an application running in a web browser.

## Typedefs

```
typedef QPair<QIcon, int> WebIconTextBannerItemIcon
```

```
typedef QPair<WebIconTextBannerItemIcon, QString> WebIconTextBannerItem
```

```
typedef QMap<QString, QString> WebCommandData
```

```
class WebAboutDialog : public QObject, public sh::ui::web::WebDialog, public sh::ui::AboutDialog  
    #include <webaboutdialog.h> Web based about dialog.
```

## Public Functions

```
WebAboutDialog ()
```

```
QVariant dialogProperty (QString dlgPropertyKey, QVariant deflt = QVariant())
```

Returns a dialog property value by key.

```
void setDialogProperty (QString dlgPropertyKey, QVariant value)
```

Sets a dialog property value for a key.

```
QVariantHash dialogResult ()
```

Returns the dialog result as hash.

The browser side of a web dialog can ‘accept’ a dialog while closing by setting a non-empty dialog result. This mechanism allows browser side dialogs to return data to the backend. It’s up to the backend to interpret the content of this hash.

If the dialog was cancelled, or is still open, the returned hash is empty. See also `wasClosed()`.

```
void triggerDialogEvent (QString name, QJsonObject data = QJsonObject())
```

Triggers a dialog event (at the browsers).

They in turn typically fetch some data again in order to refresh the ui.

```
void setCloseableByUser (bool v = true)
```

Sets if this dialog shall be directly closeable by the user by the window decoration. Set to `false` if the user has to react on this dialog by clicking on some controls in the dialog body.

This method must be called during construction.

```
bool isCloseableByUser ()
```

Returns if this dialog shall be directly closeable by the user by the window decoration.

```
QJsonObject toJson () override
```

Returns the json representation as QJsonObject.

```
qint64 dialogId ()
```

Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

```
bool isInitd ()
```

Returns if this dialog is initialized.

Must be called in main thread.

```
bool wasAccepted ()
```

Returns if this dialog was ‘accepted’ by the user (typically by clicking some ‘OK’ button).

Must be called in main thread.

void **waitClosed**()  
Wait until the user closed the dialog in some way.  
May be called in any thread.

void **close**()  
Closes the dialog.  
Must be called in main thread.

bool **wasClosed**()  
Returns if this dialog was closed.  
Must be called in main thread.

*DialogManager* \***manager**()  
Returns the *DialogManager* which hosts this dialog.

**class WebActionExecutionInfoDialog**: public *sh::ui::ActionExecutionInfoDialog*  
*#include <webactionexecutioninfodialog.h>* Web based action execution dialog.

## Public Types

**enum MessageBoxButton**  
Buttons in a message box from *ActionExecutionUserFeedback*.  
*Values:*  
**enumerator** **NONE** = 0  
**enumerator** **OK** = 1 << 0  
**enumerator** **Continue** = 1 << 1  
**enumerator** **Cancel** = 1 << 2  
**enumerator** **Retry** = 1 << 3  
**enumerator** **Yes** = 1 << 4  
**enumerator** **No** = 1 << 5

## Public Functions

**WebActionExecutionInfoDialog**(*sh::actions::ActionExecutionInfo* \**info*)  
**void setDetails**(QString *fv*, QString *fob*, QString *tv*, QString *tob*)  
Sets the item details text ('from a/foo.jpg', 'to b/foo.jpg'). .  
**void setHead**(QString *txt*)  
Sets the header text. .  
**void setProgress**(bool *isDeterminate*, quint64 *done*, quint64 *all*, QString *text*)  
Sets the progress. .  
**int messageBox**(QString *text*, QList<QString> *answers*, QString *icon* = QString(), int *defaultanswer* = -1, int *cancelanswer* = -1, QList<QString> *answericons* = QList<QString>())  
**int inputBox**(QString *text*, QList<QString> *answers*, QString \**value*, QString *icon* = QString(), int *defaultanswer* = -1, int *cancelanswer* = -1, int *valuePreselectFrom* = -1, int *valuePreselectTo* = -1)



```

int multilineInputDialog (QString text, QList<QString> answers, QString *value, QString
                           icon = QString(), int defaultanswer = -1, int cancelanswer = -1)

int simpleChooserGridform (QString text, GridformEntries *entries, QStringList answers,
                           int defaultanswer = -1, int cancelanswer = -1)

bool credentialsDialog (QString text, bool showDomain, bool showUsername, bool
                        showPassword, bool showAnonymous, bool showRemember,
                        QString *domain, QString *username, QString *password, bool
                        *isAnonymous, bool *isRemember)

bool unixPermissionsDialog (bool *userMayRead, bool *userMayWrite, bool *user-
                             MayExecute, bool *groupMayRead, bool *groupMay-
                             Write, bool *groupMayExecute, bool *othersMayRead,
                             bool *othersMayWrite, bool *othersMayExecute, bool
                             *sticky, bool *setuid, bool *setgid, QStringList users,
                             QStringList groups, QString *ownerUser, QString *own-
                             erGroup)

qint64 id ()

qint64 webts_created ()

QString details_fromverb ()

QString details_fromobject ()

QString details_toverb ()

QString details_toobject ()

QString head ()

bool progress_isDeterminate ()

quint64 progress_done ()

quint64 progress_all ()

QString progress_text ()

JsonValue userFeedbackAsJsonValue ()

bool isLogicallyVisible ()
    Returns if this dialog is logically visible (i.e. set visible by the action).

void setLogicallyVisible (bool v)
    Sets if this dialog is logically visible (i.e. set visible by the action). .

bool isBackground ()
    Returns if this dialog is currently in background mode (i.e. not visible).

void setBackground (bool v)
    Sets if this dialog is currently in background mode (i.e. not visible). .

void setForceForeground (bool v)
    Sets if this dialog is currently forced to be visible in foreground. .

QString simpleInputDialog (QString text, QString deft, int valuePreselectFrom = -1, int val-
                             uePreselectTo = -1)

int simpleMessageBox (QString text, int buttons = (int) MessageBoxButton::OK, QString
                     icon = QString(), int defaultbutton = (MessageBoxButton)0, int can-
                     celbutton = (MessageBoxButton)0)

```

### Private Functions

```
void _handleUserFeedback (QString kind, std::shared_ptr<UserFeedback> userfeedback)
void _triggerChanged ()
```

### Private Members

```
qint64 _id
QString _details_fv
QString _details_fob
QString _details_tv
QString _details_tob
QString _head
bool _progress_isDeterminate
quint64 _progress_done
quint64 _progress_all
QString _progress_text
std::shared_ptr<UserFeedback> _currentUserFeedback = 0
QMutex _currentUserFeedbackMutex
QWaitCondition _currentUserFeedbackChangedCondition
qint64 _webts_created
```

### Private Static Functions

```
QByteArray iconToBase64Src (QString icon, int sizeInPt)
QList<QVariant> iconsToBase64Srcs (QStringList icons, int sizeInPt)
```

### Friends

```
friend class WebActionManager
class UserFeedback : public QMap<QString, QVariant>
    #include <webactionexecutioninfodialog.h>
```

### Public Members

```
qint64 id = -1
bool answered = false
class WebActionExecutionInfoPanel : public sh::ui::ActionExecutionInfoPanel
    #include <webactionexecutioninfopanel.h> Web based action execution info panel.
```

## Public Functions

```

WebActionExecutionInfoPanel (sh::actions::ActionExecutionInfo *info, QColor color)
~WebActionExecutionInfoPanel ()

void setLabel (QString s)
    Sets the label text. .

void setProgress (bool isProgressDeterminate, quint64 progressDone, quint64 progressAll)
    Sets the progress. .

void setForceForeground (bool v)
    Sets if the associated action is currently forced to be visible in foreground. .

void setPanelVisible (bool v)
    Sets if the panel is visible. .

void setWidth (int width)
    Sets the panel with (in pixels). .

quint64 id ()

QString label ()

bool isProgressDeterminate ()

quint64 progressDone ()

quint64 progressAll ()

bool forceForeground ()

bool panelVisible ()

int width ()

QColor color ()

sh::actions::ActionExecutionInfo *info ()

void onClicked (std::function<void>
    > fcnQObject *owner = 0)Sets a handler for a click on the button (optionally bound to an owner
    lifetime).

void onDestroyed (std::function<void>
    > fcnQObject *owner = 0)Sets a handler for panel removal (optionally bound to an owner lifetime).

void onVisibilityChanged (std::function<void>
    > fcnQObject *owner = 0)Sets a handler for panel visibility changes (optionally bound to an owner
    lifetime).

```

## Private Functions

```

void _triggerChanged ()

```

### Private Members

```
QColor _color
sh::actions::ActionExecutionInfo *_info
qint64 _id
QString _label
bool _isProgressDeterminate
quint64 _progressDone
quint64 _progressAll
bool _forceForeground
bool _panelVisible
int _width
```

### Friends

```
friend class WebActionManager
```

```
class WebActionManager : public QObject, public sh::ui::web::WebModule
#include <webactionmanager.h> Manages sh::actions::AbstractActionItem handling,
e.g. showing them in the toolbar, executing them, and displaying their user interface.
```

### Public Functions

```
WebActionManager ()
```

```
void initialize ()
```

```
std::shared_ptr<WebActionExecutionInfoPanel> addInfoPanel (int position,
sh::actions::ActionExecutionInfo
*info, QColor color =
QColor())
```

Adds and returns a new action info panel.

```
std::shared_ptr<WebActionExecutionInfoDialog> addActionExecutionInfoDialog (sh::actions::ActionExecut
*info)
```

Adds and returns a new action info dialog.

```
void refreshToolbar (std::shared_ptr<sh::actions::ActionInstantiation> instantiation)
Refreshes the main toolbar.
```

```
bool rootCommand (WebServerEngineRequest *request, QString webadapter)
Returns the root ('index.html') content. .
```

```
void setEngine (WebServerEngine *webserver)
Assigns this web module to a sh::ui::web::WebServerEngine. .
```

```
WebServerEngine *webserver ()
Returns the sh::ui::web::WebServerEngine this web module is assigned to (may be nullptr).
```

## Public Static Functions

bool **executeCommand\_byQObjectReflection** (QObject \*o, *WebServerEngineRequest* \*request, QString command, QString justLeftside = QString())

Convenience function often used by *executeCommand()*.

For a QObject, it searches there for a slot with the name cmd\_{command}, with `/'s replaced by '\_'s (so, for the command "foo/get\_bars", it searches for slotcmd\_foo\_get\_bars`). If it finds one, it gets executed (in a worker thread). The request is considered as handled, if so.

If there is such a slot, but with appended \_\_M in name, it is called in main thread!

**Return** If it handled (answered) the command.

## Private Functions

bool **executeCommand** (*WebServerEngineRequest* \*request, QString command) **override**  
Executes command as requested by the browser, if responsible.

Read parameters, check if this module is responsible for answering, and then write an answer with request.

Implementations often use *executeCommand\_byQObjectReflection()* for convenience.

**Return** If it handled (answered) the command.

void **\_observeToolbarAction** (*sh::actions::AbstractActionItem* \*a)

std::shared\_ptr<*sh::actions::AbstractActionItem*> **resolveActionpath** (QString saction-path, QStringList \*actionpath, QList<QPair<QString, std::shared\_ptr<*sh::actions::AbstractActionItem*>> \*subactions)

void **\_triggerActionuiChanged** ()

## Private Members

QList<*WebActionExecutionInfoPanel*\*> **\_infopanel**s

QList<*WebActionExecutionInfoDialog*\*> **\_infodial**ogs

QTimer **\_triggerActionuiChangedTimer**

QTimer **\_triggerToolbarRefreshTimer**

QMap<QString, std::shared\_ptr<*sh::actions::SubmenuActionItem*>> **\_permanentToolbarActions**

### Private Slots

```
void cmd__answer_userfeedback__M(WebServerEngineRequest *request)
void cmd__execute(WebServerEngineRequest *request)
void cmd__get(WebServerEngineRequest *request)
void cmd__get_ui__M(WebServerEngineRequest *request)
void cmd__icon(WebServerEngineRequest *request)
void cmd__panel_clicked__M(WebServerEngineRequest *request)
```

### Private Static Attributes

QRegularExpression **reActionTextAccel**

### Friends

```
friend class WebActionExecutionInfoPanel
friend class WebActionExecutionInfoDialog
```

```
class WebDetailPanelManager : public QObject, public sh::ui::web::WebModule
    #include <webdetailpanelmanager.h> Manages the detail panel (typically in bottom part of main
    window).
```

### Public Functions

```
WebDetailPanelManager()
```

```
void load(QList<std::shared_ptr<sh::filesystem::FilesystemNode>> nodes)
    Loads details for a given list of nodes (typically called after they get selected).
```

```
bool rootCommand(WebServerEngineRequest *request, QString webadapter)
    Returns the root ('index.html') content. .
```

```
void setEngine(WebServerEngine *webserver)
    Assigns this web module to a sh::ui::web::WebServerEngine. .
```

```
WebServerEngine *webserver()
    Returns the sh::ui::web::WebServerEngine this web module is assigned to (may be nullptr).
```

### Public Static Functions

```
bool executeCommand_byQObjectReflection(QObject *o, WebServerEngineRequest
    *request, QString command, QString
    justLeftside = QString())
```

Convenience function often used by *executeCommand()*.

For a QObject, it searches there for a slot with the name cmd\_{command}, with ``'s replaced by '\_s (so, for the command "foo/get\_bars", it searches for slotcmd\_foo\_get\_bars`). If it finds one, it gets executed (in a worker thread). The request is considered as handled, if so.

If there is such a slot, but with appended \_\_M in name, it is called in main thread!

**Return** If it handled (answered) the command.

## Private Functions

bool **executeCommand** (*WebServerEngineRequest \*request*, QString *command*) **override**  
 Executes *command* as requested by the browser, if responsible.

Read parameters, check if this module is responsible for answering, and then write an answer with *request*.

Implementations often use *executeCommand\_byQObjectReflection()* for convenience.

**Return** If it handled (answered) the command.

## Private Members

QMutex **\_mutex**

QList<std::shared\_ptr<sh::paneldetails::PanelDetail>> **\_paneldetails**

qint64 **\_webts\_lastupdate**

## Private Slots

void **cmd\_\_get\_details** (*WebServerEngineRequest \*request*)

void **cmd\_\_get\_thumbnail** (*WebServerEngineRequest \*request*)

void **cmd\_\_link\_clicked** (*WebServerEngineRequest \*request*)

**class WebDialog**: public *sh::tools::Jsonable*

*#include <webdialog.h>* Abstract base class for a web based dialog. Typically used for also implementing some *sh::ui::Dialog*.

A web dialog is a dialog window presented to the user on browser side. A web dialog implementation consists of a backend part (a subclass of *WebDialog*) and a browser side part (basically a subclass of *shwebui.Dialog*).

For implementing a web dialog, implement a subclass of *WebDialog* (which in turn will specify the browser side part) and *sh::ui::Dialog*. Create such dialogs by means of *WebDialogManager*.

Communicating values between backend and browser typically works by *dialogProperty()* and *setDialogProperty()*, also *dialogResult()* for the dialog result after closing.

Subclassed by *sh::ui::web::WebAboutDialog*, *sh::ui::web::WebExceptionDialog*,  
*sh::ui::web::WebFilePropertyDialog*, *sh::ui::web::WebLogViewDialog*,  
*sh::ui::web::WebManageBookmarksDialog*, *sh::ui::web::WebManageProfilesDialog*,  
*sh::ui::web::WebOpenWithDialog*, *sh::ui::web::WebStoreProfileDialog*,  
*sh::ui::web::WebTuningDialog*

## Public Functions

**WebDialog** (QString *dialogClassName*)

See *WebDialogManager*.

### Parameters

- *dialogClassName*: The name of the *shwebui.Dialog* subclass which implements the web dialog on browser side.

QVariant **dialogProperty** (QString *dlgPropertyKey*, QVariant *deflt* = QVariant())

Returns a dialog property value by key.

void **setDialogProperty** (QString *dlgPropertyKey*, QVariant *value*)

Sets a dialog property value for a key.

QVariantHash **dialogResult** ()

Returns the dialog result as hash.

The browser side of a web dialog can ‘accept’ a dialog while closing by setting a non-empty dialog result. This mechanism allows browser side dialogs to return data to the backend. It’s up to the backend to interpret the content of this hash.

If the dialog was cancelled, or is still open, the returned hash is empty. See also `wasClosed()`.

void **triggerDialogEvent** (QString *name*, QJsonObject *data* = QJsonObject())

Triggers a dialog event (at the browsers).

They in turn typically fetch some data again in order to refresh the ui.

void **setCloseableByUser** (bool *v* = true)

Sets if this dialog shall be directly closeable by the user by the window decoration. Set to `false` if the user has to react on this dialog by clicking on some controls in the dialog body.

This method must be called during construction.

bool **isCloseableByUser** ()

Returns if this dialog shall be directly closeable by the user by the window decoration.

**~WebDialog** ()

QJsonObject **toJson** () **override**

Returns the json representation as QJsonObject.

## Private Members

QVariantHash **\_properties**

QString **\_dialogClassName**

qint64 **\_webts\_created**

## Friends

**friend class** WebDialogManager

**class** WebDialogManager : **public** QObject, **public** *sh::ui::web::WebModule*, **public** *sh::ui::DialogManager*  
*#include <webdialog.h>* A *DialogManager* used in Web ui.

## Public Functions

**WebDialogManager** ()

*WebServerEngine* \***webserver** ()

Return the *WebServerEngine* hosting this dialog.

bool **rootCommand** (*WebServerEngineRequest* \**request*, QString *webadapter*)

Returns the root (‘index.html’) content. .

void **setEngine** (*WebServerEngine* \**webserver*)

Assigns this web module to a *sh::ui::web::WebServerEngine*. .



std::shared\_ptr<*Dialog*> **getDialogById** (qint64 *id*)

Returns a *Dialog* by dialog id.

Must be called in main thread.

QList<std::shared\_ptr<*Dialog*>> **getAllDialogs** ()

Returns a list of all dialogs currently shown (in order of creation).

Must be called in main thread.

QList<qint64> **getAllDialogIds** ()

Returns a list of dialog ids of all dialogs currently shown (in order of creation).

Must be called in main thread.

template<class **TDlg**, typename ...**Args**>

std::shared\_ptr<*TDlg*> **createAndShowDialog** (*Args*... *args*)

Creates and shows a *Dialog* by class and constructor parameters.

Those dialogs (*TDlg*) have to implement *Dialog* (typically a subclass of it, representing a particular dialog, like *FilePropertyDialog*), and also some ui mode (e.g. qt, web) specific class (depends on that ui mode).

You typically should not have to use it outside of *MainWindow* or subclasses. The *MainWindow* interface allows to create all available dialogs in a ui mode independent way.

May be called in any thread.

**Return** The created *Dialog* instance.

## Public Static Functions

bool **executeCommand\_byQObjectReflection** (QObject \**o*, *WebServerEngineRequest* \**request*, QString *command*, QString *justLeftside* = QString())

Convenience function often used by *executeCommand()*.

For a QObject, it searches there for a slot with the name cmd\_{command}, with `/'s replaced by '\_'s (so, for the command "foo/getBars", it searches for slotcmd\_foo\_getBars). If it finds one, it gets executed (in a worker thread). The request is considered as handled, if so.

If there is such a slot, but with appended \_\_M in name, it is called in main thread!

**Return** If it handled (answered) the command.

## Private Functions

std::shared\_ptr<*Dialog*> **getDialogForRequest** (*WebServerEngineRequest* \**request*)

Returns the *WebDialog* referred to the request (i.e. its dialogId parameter).

bool **executeCommand** (*WebServerEngineRequest* \**request*, QString *command*) **override**

Executes command as requested by the browser, if responsible.

Read parameters, check if this module is responsible for answering, and then write an answer with request.

Implementations often use *executeCommand\_byQObjectReflection()* for convenience.

**Return** If it handled (answered) the command.

void **showDialog** (std::shared\_ptr<*Dialog*> *dialog*) **override**

This method implements the ui mode specific mechanism for showing a dialog.

Must be called in main thread.

### Private Slots

void **cmd\_\_change\_dialog\_property\_\_M** (*WebServerEngineRequest* \**request*)

void **cmd\_\_closed\_in\_browser\_\_M** (*WebServerEngineRequest* \**request*)

void **cmd\_\_list\_\_M** (*WebServerEngineRequest* \**request*)

**class WebExceptionDialog** : public QObject, public *sh::ui::web::WebDialog*, public *sh::ui::ExceptionDialog*

*#include <webexceptiondialog.h>* Web based exception dialog.

### Public Functions

**WebExceptionDialog** (QString *error1*, QString *error2*, QString *details*, QString *icon*, bool *mayRetry*, bool *mayClose*, bool *mayCancel*, bool *showLoglabelAndDetails*)

*ExceptionDialogResult* **answer** () **override**

Returns the answer the user has given in this dialog.

QVariant **dialogProperty** (QString *dlgPropertyKey*, QVariant *deflt* = QVariant())

Returns a dialog property value by key.

void **setDialogProperty** (QString *dlgPropertyKey*, QVariant *value*)

Sets a dialog property value for a key.

QVariantHash **dialogResult** ()

Returns the dialog result as hash.

The browser side of a web dialog can ‘accept’ a dialog while closing by setting a non-empty dialog result. This mechanism allows browser side dialogs to return data to the backend. It’s up to the backend to interpret the content of this hash.

If the dialog was cancelled, or is still open, the returned hash is empty. See also `wasClosed()`.

void **triggerDialogEvent** (QString *name*, QJsonObject *data* = QJsonObject())

Triggers a dialog event (at the browsers).

They in turn typically fetch some data again in order to refresh the ui.

void **setCloseableByUser** (bool *v* = true)

Sets if this dialog shall be directly closeable by the user by the window decoration. Set to `false` if the user has to react on this dialog by clicking on some controls in the dialog body.

This method must be called during construction.

bool **isCloseableByUser** ()

Returns if this dialog shall be directly closeable by the user by the window decoration.

QJsonObject **toJson** () **override**

Returns the json representation as QJsonObject.

QString **error1** ()

Returns the 1st error text.

QString **error2** ()

Returns the 2nd error text.

`QString details ()`  
Returns the error details.

`QString icon ()`  
Returns the icon (as name resolveable by *sh::base::IconManager*).

`bool mayRetry ()`  
Returns if it's allowed to retry.

`bool mayClose ()`  
Returns if it's allowed to just close and continue without closing Shallot.

`bool mayCancel ()`  
Returns if it's allowed to cancel, i.e. throwing a *sh::exceptions::CancelException*.

`bool showLoglabelAndDetails ()`  
Returns if the dialog shall allow to read the details.

`qint64 dialogId ()`  
Returns the dialog id.  
  
Each instance has an id unique in the complete Shallot process lifetime.  
  
Must be called in main thread.

`bool isInitiated ()`  
Returns if this dialog is initialized.  
  
Must be called in main thread.

`bool wasAccepted ()`  
Returns if this dialog was 'accepted' by the user (typically by clicking some 'OK' button).  
  
Must be called in main thread.

`void waitClosed ()`  
Wait until the user closed the dialog in some way.  
  
May be called in any thread.

`void close ()`  
Closes the dialog.  
  
Must be called in main thread.

`bool wasClosed ()`  
Returns if this dialog was closed.  
  
Must be called in main thread.

*DialogManager* \*`manager ()`  
Returns the *DialogManager* which hosts this dialog.

## Private Slots

void **cmd\_\_icon\_\_M** (*WebServerEngineRequest \*request*)

**class WebFilePropertyDialog** : public QObject, public *sh::ui::web::WebDialog*, public *sh::ui::FilePropertyDialog*  
*#include <webfilepropertydialog.h>* Web based file property dialog.

## Public Functions

void **refresh** () **override**  
Refreshes the dialog content.

*FilePropertyDialogTabTableView* \***createTabViewTable** () **override**  
Creates a new tab view table sub-widget.

*FilePropertyDialogTabTextView* \***createTabViewText** () **override**  
Creates a new tab view text sub-widget.

*FilePropertyDialogTabIconTextBannerView* \***createTabViewIconTextBanner** () **override**  
Creates a new tab view icon text banner sub-widget.

**WebFilePropertyDialog** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>>  
*nodes*)

**~WebFilePropertyDialog** ()

QVariant **dialogProperty** (QString *dlgPropertyKey*, QVariant *deflt* = QVariant())  
Returns a dialog property value by key.

void **setDialogProperty** (QString *dlgPropertyKey*, QVariant *value*)  
Sets a dialog property value for a key.

QVariantHash **dialogResult** ()  
Returns the dialog result as hash.

The browser side of a web dialog can ‘accept’ a dialog while closing by setting a non-empty dialog result. This mechanism allows browser side dialogs to return data to the backend. It’s up to the backend to interpret the content of this hash.

If the dialog was cancelled, or is still open, the returned hash is empty. See also wasClosed().

void **triggerDialogEvent** (QString *name*, QJsonObject *data* = QJsonObject())  
Triggers a dialog event (at the browsers).

They in turn typically fetch some data again in order to refresh the ui.

void **setCloseableByUser** (bool *v* = true)  
Sets if this dialog shall be directly closeable by the user by the window decoration. Set to false if the user has to react on this dialog by clicking on some controls in the dialog body.

This method must be called during construction.

bool **isCloseableByUser** ()  
Returns if this dialog shall be directly closeable by the user by the window decoration.

QJsonObject **toJson** () **override**  
Returns the json representation as QJsonObject.

QList<std::shared\_ptr<*FilePropertyDialogTab*>> **tabs** ()  
Returns a list of all tabs.

*sh::ui::FilePropertyDialogTabActionsView* \***widgetAt** (std::shared\_ptr<*FilePropertyDialogTab*>  
*tab*, int *i*)

Returns the widget from a tab at a certain index.

int **widgetCount** (std::shared\_ptr<*FilePropertyDialogTab*> *tab*)

Returns the number of widgets in a tab.

qint64 **dialogId** ()

Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

bool **isInited** ()

Returns if this dialog is initialized.

Must be called in main thread.

bool **wasAccepted** ()

Returns if this dialog was ‘accepted’ by the user (typically by clicking some ‘OK’ button).

Must be called in main thread.

void **waitClosed** ()

Wait until the user closed the dialog in some way.

May be called in any thread.

void **close** ()

Closes the dialog.

Must be called in main thread.

bool **wasClosed** ()

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \***manager** ()

Returns the *DialogManager* which hosts this dialog.

## Public Static Functions

void **addTabFactory** (int *i*, std::shared\_ptr<*FilePropertyDialogTabFactory*> *factory*)

Adds a *FilePropertyDialogTabFactory* so the Properties dialogs show its content.

See also the REGISTER\_FILEPROPERTYDIALOGTAB macro.

### Parameters

- *i*: An index which controls the display order. Use one of the REGISTER\_FILEPROPERTYDIALOGTAB\_INDEX\_\* values in *FilePropertyDialogTabFactory* as base values.

### Private Functions

```
void tabwidgetTableSelect (int cookie, int tabidx, int widgetidx, QList<int> lsel)  
void triggerAction (int cookie, int tabidx, int widgetidx, int btnidx)  
QJsonArray getProperties ()
```

### Private Members

```
int _cookie = 0  
QTimer _refreshTimer  
QList<WebFilePropertyDialogChild*> _childs
```

### Private Slots

```
void cmd__get_properties__M (WebServerEngineRequest *request)  
void cmd__refresh__M (WebServerEngineRequest *request)  
void cmd__trigger_action__M (WebServerEngineRequest *request)  
void cmd__tabwidget_table_select__M (WebServerEngineRequest *request)  
class WebFilePropertyDialogChild  
    Subclassed by sh::ui::web::WebFilePropertyDialog::WebFilePropertyDialogTabActionsView,  
    sh::ui::web::WebFilePropertyDialog::WebFilePropertyDialogTabIconTextBannerView,  
    sh::ui::web::WebFilePropertyDialog::WebFilePropertyDialogTabTableView,  
    sh::ui::web::WebFilePropertyDialog::WebFilePropertyDialogTabTextView
```

### Public Functions

```
WebFilePropertyDialogChild (WebFilePropertyDialog *dlg)  
~WebFilePropertyDialogChild ()  
class WebFilePropertyDialogTabActionsView: public sh::ui::FilePropertyDialogTabActionsView, public
```

### Public Functions

```
WebFilePropertyDialogTabActionsView (WebFilePropertyDialog *dlg)  
void setVisible (bool v) override  
    Sets the view visible or hidden. .  
void setContent (FilePropertyDialogTabViewContent *cnt) override  
    Sets the main widget. .  
void setButtons (QList<QString> buttons) override  
    Sets the list of buttons. .  
void setLabelText (QString text)  
QJsonObject toJson () override  
    Returns the json representation as QJsonObject.
```

*FilePropertyDialogTabViewContent* \***content** ()

The main widget.

QList<QString> **buttons** ()

The list of buttons.

void **onButtonTriggered** (std::function<void> int i

> *fct*, QObject \**owner* = 0) Sets a handler for a click on one of the buttons (optionally bound to an owner lifetime).

## Private Members

bool **\_visible** = false

QString **\_labelText**

## Friends

**friend class** WebFilePropertyDialog

**class** WebFilePropertyDialogTabIconTextBannerView : public *sh::ui::FilePropertyDialogTabIconTextBannerView*

## Public Functions

**WebFilePropertyDialogTabIconTextBannerView** (*WebFilePropertyDialog*  
\**dlg*)

void **clear** () **override**

Clears the contents.

void **insertIcon** (QIcon *icon*, int *sizePt* = 24) **override**

Adds an icon.

void **insertText** (QString *text*) **override**

Adds a text.

QJsonObject **toJson** () **override**

Returns the json representation as QJsonObject.

## Private Members

QList<*WebIconTextBannerItem*> **\_content**

**class** WebFilePropertyDialogTabTableView : public *sh::ui::FilePropertyDialogTabTableView*, public

## Public Types

**typedef** QPair<int, int> **ItemIndex**

### Public Functions

**WebFilePropertyDialogTabTableView** (*WebFilePropertyDialog \*dlg*)

void **setContent** (QList<QPair<QString, QString>> *content*) **override**  
Sets the content.

QList<*ItemIndex*> **getSelectedItems** () **override**  
Returns the selected cells.

QVariant **getItemValue** (int *r*, int *c*) **override**  
Returns a value of a cell.

QJsonObject **toJson** () **override**  
Returns the json representation as QJsonObject.

void **setSelectedItems** (QList<int> *selitms*)

### Private Members

QList<QPair<QString, QString>> **\_content**

QList<int> **\_selectedItems**

**class WebFilePropertyDialogTabTextView**: public *sh::ui::FilePropertyDialogTabTextView*, public

### Public Functions

**WebFilePropertyDialogTabTextView** (*WebFilePropertyDialog \*dlg*)

void **setContent** (QString *content*) **override**  
Sets the content.

QJsonObject **toJson** () **override**  
Returns the json representation as QJsonObject.

### Private Members

QString **\_content**

**class WebFileView**: public *sh::ui::FileView*  
*#include <webfileview.h>* Web based file view.

### Public Functions

**WebFileView** (*WebFileViewManager \*manager*)

*sh::ui::ColumnDimensions* \***columnDimensions** () **override**  
Returns the column dimensions handler. .

*sh::tools::HistoryTracker*<std::shared\_ptr<const *sh::filesystem::Eurl*>> \***historyTracker** () **override**  
Returns the history tracker. .

*FileViewMode* **viewmode** () **override**  
Returns the view mode (icons, list, ... ?). .



```

void setViewmode (FileViewMode m) override
    Sets the view mode. .

int index () override
    Returns the position of this file view within the panel. .

void gotoDir (std::shared_ptr<sh::filesystem::FilesystemNode> n) override
    Jumps to a new current directory.

    Implementations have to call the base class implementation inside.

sh::filesystem::SizeFormatting getSizeFormattingMode () override
    Returns the mode how file sizes are formatted for displaying. .

void setSizeFormattingMode (sh::filesystem::SizeFormatting mode) override
    Sets the mode how file sizes are formatted for displaying. .

bool hiddenFilesVisible () override
    Returns if hidden files are visible. .

void setHiddenFilesVisible (bool v) override
    Sets the visibility of hidden files. .

int iconDimension () override
    Returns the icon size (in pixels). .

void setIconDimension (int v) override
    Sets the icon size (in pixels). .

void setSort (int column, Qt::SortOrder order) override
    Sets how to sort this view. .

int sortColumn () override
    Returns the index of the current sort column. .

Qt::SortOrder sortOrder () override
    Returns the current sort order. .

QList<std::shared_ptr<sh::filesystem::FilesystemNode>> selectedNodes () override
    Returns the nodes which the user has selected in this fileview. .

void setSelection (const QList<std::shared_ptr<sh::filesystem::FilesystemNode>>
                  nodes) override
    Sets the node selection of this fileview. .

QList<std::shared_ptr<sh::filesystem::FilesystemNode>> getAllVisibleNodes ()
                                                                    override
    Returns a list of all nodes currently listed in this file view. .

qint64 id ()
    Returns the unique identifier.

std::shared_ptr<sh::filesystem::FilesystemModelFileviewProxy> model ()
    Returns the filesystem model bound to this view.

std::shared_ptr<sh::filesystem::FilesystemNode> node ()
    Returns the current directory.

void reload (bool skipModel = false)
    Reloads the data inside this file view.

sh::filesystem::FilesystemModelFileviewProxy *filemodel ()
    Returns the filesystem model for this view. .

```

```
void setFilemodel (sh::filesystem::FilesystemModelFileviewProxy *model)  
    Sets the filesystem model for this view. .
```

## Signals

```
void currentNodeChanged ()  
void modelChanged ()  
void selectionChanged ()  
    Triggered when the selection have changed.  
void viewOptionChanged ()  
    Triggered when some view options have changed.
```

## Private Functions

```
void _set_model ()  
void setSelectedNode (std::shared_ptr<sh::filesystem::FilesystemNode> node)  
void setCheckedNodes (QList<std::shared_ptr<sh::filesystem::FilesystemNode>> nodes)
```

## Private Members

```
std::shared_ptr<sh::filesystem::FilesystemModelFileviewProxy> _model  
qint64 _id  
FileViewMode _viewmode = FileViewMode::ListView  
sh::filesystem::SizeFormatting _sizeformatting = sh::filesystem::SizeFormatting::SizeFormattingModePrefixes  
bool _hiddenFilesVisible = false  
int _iconDimension = 20  
int _sortColumn = 0  
Qt::SortOrder _sortOrder = Qt::SortOrder::AscendingOrder  
std::shared_ptr<sh::filesystem::FilesystemNode> _selectedNode  
QList<std::shared_ptr<sh::filesystem::FilesystemNode>> _checkedNodes  
WebFileViewManager *_manager
```

## Friends

```
friend class WebFileViewManager  
class WebFileViewManager : public QObject, public sh::ui::web::WebModule  
    #include <webfileview.h> Manages WebFileView instances.
```

## Public Functions

bool **executeCommand** (*WebServerEngineRequest* \*request, QString command) **override**  
 Executes command as requested by the browser, if responsible.

Read parameters, check if this module is responsible for answering, and then write an answer with request.

Implementations often use *executeCommand\_byQObjectReflection()* for convenience.

**Return** If it handled (answered) the command.

std::shared\_ptr<*WebFileView*> **addFileView** (bool reinitialize)

void **setCurrentFileViewByIndex** (int idx)

*sh::ui::web::WebFileView* \***currentFileView** ()

int **fileViewCount** ()

void **removeFileView** (int i)

*sh::ui::web::WebFileView* \***getFileView** (int i)

int **getFileViewIndex** (*sh::ui::FileView* \*fileview)

bool **rootCommand** (*WebServerEngineRequest* \*request, QString webadapter)  
 Returns the root ('index.html') content. .

void **setEngine** (*WebServerEngine* \*webserver)  
 Assigns this web module to a *sh::ui::web::WebServerEngine*. .

*WebServerEngine* \***webserver** ()  
 Returns the *sh::ui::web::WebServerEngine* this web module is assigned to (may be nullptr).

## Signals

void **activeSelectionChanged** ()

void **fileViewOptionsChanged** (int i)

void **fileViewCountChanged** ()

void **currentFileViewChanged** ()

## Public Static Functions

bool **executeCommand\_byQObjectReflection** (QObject \*o, *WebServerEngineRequest* \*request, QString command, QString justLeftside = QString())

Convenience function often used by *executeCommand()*.

For a QObject, it searches there for a slot with the name cmd\_{command}, with ``'`'s replaced by '\_'s (so, for the command "foo/getBars", it searches for slotcmd\_foo\_getBars). If it finds one, it gets executed (in a worker thread). The request is considered as handled, if so.

If there is such a slot, but with appended \_\_M in name, it is called in main thread!

**Return** If it handled (answered) the command.

### Private Members

QList<std::shared\_ptr<WebFileView>> **\_fileviews**

int **\_icurrentfileview** = 0

### Private Slots

void **cmd\_\_checkednodes\_changed\_\_M**(WebServerEngineRequest \*request)

void **cmd\_\_focus\_current\_\_M**(WebServerEngineRequest \*request)

void **cmd\_\_get\_\_M**(WebServerEngineRequest \*request)

void **cmd\_\_list\_view\_items\_\_M**(WebServerEngineRequest \*request)

void **cmd\_\_node\_activated\_\_M**(WebServerEngineRequest \*request)

void **cmd\_\_selection\_changed\_\_M**(WebServerEngineRequest \*request)

**class WebI18n**

*#include <webi18n.h>* Web related helpers for internationalization/translations.

### Public Static Functions

QString **get** ()

Returns the Javascript representation for translated strings (in current ui language).

**class WebLogViewDialog**: public QObject, public *sh::ui::web::WebDialog*, public *sh::ui::LogViewDialog*

*#include <weblogviewdialog.h>* Web based log view dialog.

### Public Functions

**WebLogViewDialog** (QString *headtext*, QString *subheadtxt*, *sh::base::LogSeverity* *minseverity*)

QVariant **dialogProperty** (QString *dlgPropertyKey*, QVariant *deflt* = QVariant())

Returns a dialog property value by key.

void **setDialogProperty** (QString *dlgPropertyKey*, QVariant *value*)

Sets a dialog property value for a key.

QVariantHash **dialogResult** ()

Returns the dialog result as hash.

The browser side of a web dialog can ‘accept’ a dialog while closing by setting a non-empty dialog result. This mechanism allows browser side dialogs to return data to the backend. It’s up to the backend to interpret the content of this hash.

If the dialog was cancelled, or is still open, the returned hash is empty. See also wasClosed().

void **triggerDialogEvent** (QString *name*, QJsonObject *data* = QJsonObject())

Triggers a dialog event (at the browsers).

They in turn typically fetch some data again in order to refresh the ui.

void **setCloseableByUser** (bool *v* = true)

Sets if this dialog shall be directly closeable by the user by the window decoration. Set to `false` if the user has to react on this dialog by clicking on some controls in the dialog body.

This method must be called during construction.

bool **isCloseableByUser** ()

Returns if this dialog shall be directly closeable by the user by the window decoration.

QJsonObject **toJson** () **override**

Returns the json representation as QJsonObject.

QString **headtext** ()

Returns the header text.

QString **subheadtxt** ()

Returns the 2nd header text.

*sh::base::LogSeverity* **minimumSeverity** ()

Returns the minimum severity.

qint64 **dialogId** ()

Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

bool **isInitd** ()

Returns if this dialog is initialized.

Must be called in main thread.

bool **wasAccepted** ()

Returns if this dialog was 'accepted' by the user (typically by clicking some 'OK' button).

Must be called in main thread.

void **waitClosed** ()

Wait until the user closed the dialog in some way.

May be called in any thread.

void **close** ()

Closes the dialog.

Must be called in main thread.

bool **wasClosed** ()

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* **\*manager** ()

Returns the *DialogManager* which hosts this dialog.

## Private Slots

void **cmd\_\_get\_messages** (*WebServerEngineRequest* \*request)

**class WebMainWindow**: public QObject, public *sh::ui::MainWindow*, public *sh::ui::web::WebModule*  
*#include <webmainwindow.h>* The web main window implementation.

## Public Functions

**WebMainWindow** (QString *dispatcherUrl*, QByteArray *wtoken*)

**~WebMainWindow** ()

*WebServerEngine* \***webserver** ()

Returns the *WebServerEngine* instance for this main window.

void **initialize** () **override**

Initializes this main window. .

std::shared\_ptr<*DialogManager*> **dialogManager** () **override**

Returns the *DialogManager* which creates and drives *Dialogs* for this main window.

You typically should not need to use it directly.

int **fileViewCount** () **override**

Returns the current number of file views. .

void **addFileView** (bool *reinitialize* = false) **override**

Adds a new file view. .

void **removeFileView** (int *i*) **override**

Removes a file view. .

*sh::ui::FileView* \***currentFileView** () **override**

Returns the file view which is currently active (i.e. focussed). .

*sh::ui::FileView* \***getFileView** (int *i*) **override**

Returns a file view by position index. .

void **jumpToNode** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> *node*) **override**

Let the current view jump to another location. .

std::shared\_ptr<*sh::filesystem::FilesystemNode*> **currentDirectory** () **override**

Gets the *sh::filesystem::FilesystemNode* selected in the current view. .

void **setTitlePattern** (QString *value*) **override**

Sets the window title pattern. .

void **setTreeVisible** (bool *v*) **override**

Sets the visibility of the directory tree. .

void **setTreeSticky** (bool *v*) **override**

Sets if the directory tree should follow the active file view. .

void **setFileDetailsPanelVisible** (bool *v*) **override**

Sets the visibility of the file details panel. .

std::shared\_ptr<*sh::ui::ActionExecutionInfoPanel*> **addInfoPanel** (int *position*,  
*sh::actions::ActionExecutionInfo*  
\**info*, QColor *color* =  
QColor()) **override**

Creates a new action execution panel.

Let this smart pointer die for removing it.

void **jumpbarSetTextMode** () **override**

Sets the jumpbar to text input mode. .

void **jumpbarSetButtonMode** () **override**

Sets the jumpbar to button mode. .

```

bool jumpbarIsTextMode () override
    Returns if the jumpbar is in text input mode. .

QString toolbarPosition () override
    Returns the current toolbar position. .

void setToolBarPosition (QString pos) override
    Sets the toolbar position. .

QString detailPanelPosition () override
    Returns the current detail panel position. .

void setDetailPanelPosition (QString pos) override
    Sets the detail panel position. .

std::shared_ptr<AboutDialog> showAboutDialog () override
    Shows and returns an 'About' dialog. .

std::shared_ptr<ManageProfilesDialog> showManageProfilesDialog () override
    Shows and returns a 'Manage Saved Settings' dialog. .

std::shared_ptr<OpenWithDialog> showOpenWithDialog (std::shared_ptr<sh::filesystem::FilesystemNode>
                                                    node,
                                                    QList<std::shared_ptr<sh::tools::filetypes::OpenMethod>
                                                    openMethods) override
    Shows and returns a 'Open With' dialog. .

std::shared_ptr<StoreProfileDialog> showStoreProfileDialog (std::shared_ptr<const
                                                            sh::filesystem::Eurl>
                                                            eurl) override
    Shows and returns a 'Save Settings' dialog. .

std::shared_ptr<TuningDialog> showTuningDialog () override
    Shows and returns a 'Tuning' dialog. .

std::shared_ptr<LogViewDialog> showLogViewDialog (QString headtext = QString(),
                                                    QString subheadtxt = QString(),
                                                    sh::base::LogSeverity minseverity =
                                                    sh::base::LogSeverity::_DEFAULT)
                                                    override
    Shows and returns a 'Log' dialog. .

std::shared_ptr<ManageBookmarksDialog> showManageBookmarksDialog ()
                                                    override
    Shows and returns a 'Manage Bookmarks' dialog. .

std::shared_ptr<FilePropertyDialog> showFilePropertyDialog (QList<std::shared_ptr<sh::filesystem::FilesystemNode>
                                                            nodes) override
    Shows and returns a 'File Properties' dialog. .

std::shared_ptr<ActionExecutionInfoDialog> createActionExecutionInfoDialog (sh::actions::ActionExecutionInfo
                                                                            *info)
                                                                            override
    Creates an action execution dialog. .

void handleCloseAppRejected (QString rejectmsg) override
    React on a rejected application close request (with some feedback message). .

void _initialize (sh::base::SingletonInitializer *singletonInitializer)
    Initializes the main window. For custom initialization logic, see MainWindow.initialize. .

void fileViewsReloadAll ()
    Reload all content in each file view.

```

```
void onFileViewOptionsChanged (std::function<void>) int
    > fcn, QObject *owner = 0 Sets a handler for some view options in a file view changed (optionally
    bound to an owner lifetime).

void onCurrentFileViewChanged (std::function<void>)
    > fcn QObject *owner = 0 Sets a handler for the currently active file view changed (optionally
    bound to an owner lifetime).

void onFileViewCountChanged (std::function<void>)
    > fcn QObject *owner = 0 Sets a handler for the number of file views changed (optionally bound
    to an owner lifetime).

void onCurrentDirectoryChanged (std::function<void>)
    > fcn QObject *owner = 0 Sets a handler for the current directory in the active file view changed
    (optionally bound to an owner lifetime).

void onGlobalViewOptionsChanged (std::function<void>)
    > fcn QObject *owner = 0 Sets a handler for some global view options changed (optionally bound
    to an owner lifetime).

void onCurrentProfileChanged (std::function<void>)
    > fcn QObject *owner = 0 Sets a handler for the current profile changed (optionally bound to an
    owner lifetime).

void jumpToEurl (std::shared_ptr<const sh::filesystem::Eurl> eurl)
    Let the current view jump to another location. .

QString titlePattern ()
    Returns the window title pattern.

void setCurrentProfile (QString profile)
    Sets the current profile. .

QString currentProfile ()
    Returns the current profile.

void reloadProfile ()
    Reloads the profile data.

bool treeVisible ()
    Returns the visibility of the directory tree.

bool treeSticky ()
    Returns if the directory tree follows the active file view.

bool fileDetailsPanelVisible ()
    Returns the visibility of the file details panel.

std::shared_ptr<sh::ui::ActionExecutionInfoPanel> addInfoPanel (sh::actions::ActionExecutionInfo
    *info, QColor color =
    QColor())
    Creates a new action execution panel.
    Let this smart pointer die for removing it.

std::shared_ptr<ActionExecutionInfoPanel> showErrorPanel ()
    Shows an error panel.

void _closeDirectly ()
    Directly begin application shutdown without checks.

bool closeApp ()
    Check if the application may shut down now, and if so, initiate it.
```



bool **isCloseable** (QString \*msg = nullptr)  
 Check if the application may shut down now.

void **handleClosed** ()  
 Do some cleanup steps just when closing starts. Implementations must call base implementation!

bool **isClosing** ()  
 Returns if the application is currently closing.

bool **rootCommand** (WebServerEngineRequest \*request, QString webadapter)  
 Returns the root ('index.html') content. .

void **setEngine** (WebServerEngine \*webserver)  
 Assigns this web module to a *sh::ui::web::WebServerEngine*. .

## Public Static Functions

bool **tryCreateMainWindow** (MainWindow \*&mainWindow)  
*WebMainWindow* \*mainWindow ()

void **openUrlOnDesktop** (QUrl url)  
 Opens a url on the user desktop, typically in the browser.  
 Note: It obeys the 'openbrowser' ui parameter.

void **setMainWindow** (MainWindow \*mainWindow)  
 Sets the main window instance. .

bool **isReady** ()  
 Returns if this process ui is ready (i.e. is created or runs headless).

bool **runsHeadless** ()  
 Returns if this process runs headless, i.e. without any ui, just for a background process.

void **\_closeDirectly** (sh::base::SingletonInitializer \*singletonInitializer)

QString **uiMode** ()  
 Returns the UI mode (e.g. qt, web).

bool **executeCommand\_byQObjectReflection** (QObject \*o, WebServerEngineRequest \*request, QString command, QString justLeftside = QString())  
 Convenience function often used by *executeCommand()*.  
 For a QObject, it searches there for a slot with the name cmd\_{command}, with '/'s replaced by '\_'s (so, for the command "foo/getBars", it searches for slotcmd\_foo\_getBars). If it finds one, it gets executed (in a worker thread). The request is considered as handled, if so.  
 If there is such a slot, but with appended \_\_M in name, it is called in main thread!  
**Return** If it handled (answered) the command.

### Private Functions

```
std::shared_ptr<sh::ui::web::WebActionManager> actionManager ()  
void setCloseable (bool closeable, QString message = QString()) override  
    Sets the closeable state and message.  
void setCurrentEurlByNode (std::shared_ptr<sh::filesystem::FilesystemNode> node)
```

### Private Members

```
bool _treeSticky = true  
bool _detailsPanelVisible = true  
QString _currentProfile  
WebServerEngine * _webserver  
std::shared_ptr<sh::filesystem::FilesystemNode> _currentDirectory = 0  
std::shared_ptr<WebActionManager> _actionManager  
std::shared_ptr<WebDetailPanelManager> _detailPanelManager  
std::shared_ptr<WebDialogManager> _dialogManager  
std::shared_ptr<WebFileViewManager> _fileViewManager  
QString _lastCloseableState
```

### Private Slots

```
void cmd_filesystem_get_icon (WebServerEngineRequest *request)  
void cmd_filesystem_list_items (WebServerEngineRequest *request)  
void cmd_log_as_text (WebServerEngineRequest *request)  
void cmd_log_log_message (WebServerEngineRequest *request)  
void cmd_mainwindow_set_current_directory (WebServerEngineRequest *request) *re-  
void cmd_mainwindow_shallot_icon (WebServerEngineRequest *request)  
void cmd_mainwindow_show_logview (WebServerEngineRequest *request)
```

### Private Static Attributes

```
WebMainWindow * _webMainWindow = nullptr
```

## Friends

```
friend class WebFileViewManager
friend class WebActionExecutionInfoPanel
friend class WebActionExecutionInfoDialog
friend class WebDetailPanelManager
```

```
class WebManageBookmarksDialog : public QObject, public sh::ui::web::WebDialog, public sh::ui::Manager
#include <webmanagebookmarksdialog.h> Web based manage bookmarks dialog.
```

## Public Functions

```
WebManageBookmarksDialog ()
```

```
QVariant dialogProperty (QString dlgPropertyKey, QVariant deflt = QVariant())
Returns a dialog property value by key.
```

```
void setDialogProperty (QString dlgPropertyKey, QVariant value)
Sets a dialog property value for a key.
```

```
QVariantHash dialogResult ()
Returns the dialog result as hash.
```

The browser side of a web dialog can ‘accept’ a dialog while closing by setting a non-empty dialog result. This mechanism allows browser side dialogs to return data to the backend. It’s up to the backend to interpret the content of this hash.

If the dialog was cancelled, or is still open, the returned hash is empty. See also wasClosed().

```
void triggerDialogEvent (QString name, QJsonObject data = QJsonObject())
Triggers a dialog event (at the browsers).
```

They in turn typically fetch some data again in order to refresh the ui.

```
void setCloseableByUser (bool v = true)
Sets if this dialog shall be directly closeable by the user by the window decoration. Set to false
if the user has to react on this dialog by clicking on some controls in the dialog body.
```

This method must be called during construction.

```
bool isCloseableByUser ()
Returns if this dialog shall be directly closeable by the user by the window decoration.
```

```
QJsonObject toJson () override
Returns the json representation as QJsonObject.
```

```
qint64 dialogId ()
Returns the dialog id.
```

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

```
bool isInitiated ()
Returns if this dialog is initialized.
```

Must be called in main thread.

```
bool wasAccepted ()
Returns if this dialog was ‘accepted’ by the user (typically by clicking some ‘OK’ button).
```

Must be called in main thread.

void **waitClosed** ()

Wait until the user closed the dialog in some way.

May be called in any thread.

void **close** ()

Closes the dialog.

Must be called in main thread.

bool **wasClosed** ()

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* **\*manager** ()

Returns the *DialogManager* which hosts this dialog.

### Private Slots

void **cmd\_\_add\_bookmark\_\_M** (*WebServerEngineRequest* \*request)

void **cmd\_\_delete\_bookmark** (*WebServerEngineRequest* \*request)

void **cmd\_\_get** (*WebServerEngineRequest* \*request)

void **cmd\_\_move\_bookmark\_down** (*WebServerEngineRequest* \*request)

void **cmd\_\_move\_bookmark\_to\_folder** (*WebServerEngineRequest* \*request)

void **cmd\_\_move\_bookmark\_up** (*WebServerEngineRequest* \*request)

void **cmd\_\_store** (*WebServerEngineRequest* \*request)

**class** **WebManageProfilesDialog** : **public** **QObject**, **public** *sh::ui::web::WebDialog*, **public** *sh::ui::ManageProfilesDialog*  
*#include <webmanageprofilesdialog.h>* Web based manage saved settings dialog.

### Public Functions

**WebManageProfilesDialog** ()

**QVariant dialogProperty** (**QString** *dlgPropertyKey*, **QVariant** *deflt* = **QVariant**())

Returns a dialog property value by key.

void **setDialogProperty** (**QString** *dlgPropertyKey*, **QVariant** *value*)

Sets a dialog property value for a key.

**QVariantHash dialogResult** ()

Returns the dialog result as hash.

The browser side of a web dialog can ‘accept’ a dialog while closing by setting a non-empty dialog result. This mechanism allows browser side dialogs to return data to the backend. It’s up to the backend to interpret the content of this hash.

If the dialog was cancelled, or is still open, the returned hash is empty. See also `wasClosed()`.

void **triggerDialogEvent** (**QString** *name*, **QJsonObject** *data* = **QJsonObject**())

Triggers a dialog event (at the browsers).

They in turn typically fetch some data again in order to refresh the ui.

void **setCloseableByUser** (bool *v* = true)

Sets if this dialog shall be directly closeable by the user by the window decoration. Set to `false` if the user has to react on this dialog by clicking on some controls in the dialog body.

This method must be called during construction.

bool **isCloseableByUser** ()

Returns if this dialog shall be directly closeable by the user by the window decoration.

QObject **toJson** () **override**

Returns the json representation as QObject.

qint64 **dialogId** ()

Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

bool **isInitiated** ()

Returns if this dialog is initialized.

Must be called in main thread.

bool **wasAccepted** ()

Returns if this dialog was 'accepted' by the user (typically by clicking some 'OK' button).

Must be called in main thread.

void **waitClosed** ()

Wait until the user closed the dialog in some way.

May be called in any thread.

void **close** ()

Closes the dialog.

Must be called in main thread.

bool **wasClosed** ()

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \***manager** ()

Returns the *DialogManager* which hosts this dialog.

## Private Slots

void **cmd\_\_get\_\_M** (*WebServerEngineRequest* \**request*)

void **cmd\_\_icon** (*WebServerEngineRequest* \**request*)

void **cmd\_\_remove\_node\_\_M** (*WebServerEngineRequest* \**request*)

void **cmd\_\_remove\_profile\_\_M** (*WebServerEngineRequest* \**request*)

**class WebModule**

*#include <webmodule.h>* Base class for modules, which provide some functionality on top of a web server engine.

Subclassed by *sh::ui::web::WebActionManager*, *sh::ui::web::WebDetailPanelManager*,  
*sh::ui::web::WebDialogManager*, *sh::ui::web::WebFileViewManager*,

*sh::ui::web::WebMainWindow,*  
*sh::ui::web::WebServerEngineMainModule*

*sh::ui::web::WebServerEngineDispatcher,*

## Public Functions

**WebModule** ()

**~WebModule** ()

bool **executeCommand** (*WebServerEngineRequest* \*request, QString command)

Executes command as requested by the browser, if responsible.

Read parameters, check if this module is responsible for answering, and then write an answer with request.

Implementations often use *executeCommand\_byQObjectReflection()* for convenience.

**Return** If it handled (answered) the command.

bool **rootCommand** (*WebServerEngineRequest* \*request, QString webadapter)

Returns the root ('index.html') content. .

void **setEngine** (*WebServerEngine* \*webserver)

Assigns this web module to a *sh::ui::web::WebServerEngine*. .

*WebServerEngine* \***webserver** ()

Returns the *sh::ui::web::WebServerEngine* this web module is assigned to (may be nullptr).

## Public Static Functions

bool **executeCommand\_byQObjectReflection** (QObject \*o, *WebServerEngineRequest* \*request, QString command, QString justLeftside = QString())

Convenience function often used by *executeCommand()*.

For a QObject, it searches there for a slot with the name cmd\_{command}, with ``'s replaced by '\_'s (so, for the command"foo/get\_bars", it searches for slotcmd\_foo\_get\_bars`). If it finds one, it gets executed (in a worker thread). The request is considered as handled, if so.

If there is such a slot, but with appended \_\_M in name, it is called in main thread!

**Return** If it handled (answered) the command.

## Private Members

*WebServerEngine* \*\_**webserver** = nullptr

**class WebOpenWithDialog**: public QObject, public *sh::ui::web::WebDialog*, public *sh::ui::OpenWithDialog*  
#include <webopenwithdialog.h> Web based open with dialog.

## Public Functions

**WebOpenWithDialog** (std::shared\_ptr<sh::filesystem::FileSystemNode> *node*,  
 QList<std::shared\_ptr<sh::tools::filetypes::OpenMethod>> *open-  
 Methods*)

std::shared\_ptr<sh::tools::filetypes::OpenMethod> **chosenMethod** () **override**  
 Returns the chosen open-method (program for opening the file).

bool **rememberForMimetype** () **override**  
 Returns if the chosen method is checked to be remembered for the same mime-type in the future.

std::shared\_ptr<const sh::filesystem::Eurl> **rememberForDirectory** () **override**  
 If it's checked for the chosen method to be remembered for some subdirectory in the future, it  
 returns the Eurl of this subdirectory, otherwise nullptr.

bool **rememberForFile** () **override**  
 Returns if the chosen method is checked to be remembered for the same file in the future.

QVariant **dialogProperty** (QString *dlgPropertyKey*, QVariant *deflt* = QVariant())  
 Returns a dialog property value by key.

void **setDialogProperty** (QString *dlgPropertyKey*, QVariant *value*)  
 Sets a dialog property value for a key.

QVariantHash **dialogResult** ()  
 Returns the dialog result as hash.

The browser side of a web dialog can 'accept' a dialog while closing by setting a non-empty  
 dialog result. This mechanism allows browser side dialogs to return data to the backend. It's up  
 to the backend to interpret the content of this hash.

If the dialog was cancelled, or is still open, the returned hash is empty. See also wasClosed().

void **triggerDialogEvent** (QString *name*, QJsonObject *data* = QJsonObject())  
 Triggers a dialog event (at the browsers).

They in turn typically fetch some data again in order to refresh the ui.

void **setCloseableByUser** (bool *v* = true)  
 Sets if this dialog shall be directly closeable by the user by the window decoration. Set to `false`  
 if the user has to react on this dialog by clicking on some controls in the dialog body.

This method must be called during construction.

bool **isCloseableByUser** ()  
 Returns if this dialog shall be directly closeable by the user by the window decoration.

QJsonObject **toJson** () **override**  
 Returns the json representation as QJsonObject.

std::shared\_ptr<sh::filesystem::FileSystemNode> **node** ()  
 Returns the file node which is later to be opened.

QList<std::shared\_ptr<tools::filetypes::OpenMethod>> **openMethods** ()  
 Returns the open-methods (programs for opening the file) to present for choice.

qint64 **dialogId** ()  
 Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

bool **isInited** ()

Returns if this dialog is initialized.

Must be called in main thread.

bool **wasAccepted** ()

Returns if this dialog was 'accepted' by the user (typically by clicking some 'OK' button).

Must be called in main thread.

void **waitClosed** ()

Wait until the user closed the dialog in some way.

May be called in any thread.

void **close** ()

Closes the dialog.

Must be called in main thread.

bool **wasClosed** ()

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \***manager** ()

Returns the *DialogManager* which hosts this dialog.

### Private Slots

void **cmd\_\_check\_another\_ancestor\_\_M** (*WebServerEngineRequest* \*request)

void **cmd\_\_check\_custom\_opener\_\_M** (*WebServerEngineRequest* \*request)

void **cmd\_\_open\_method\_icon\_\_M** (*WebServerEngineRequest* \*request)

void **cmd\_\_open\_methods\_\_M** (*WebServerEngineRequest* \*request)

**class WebServerEngine**

*#include <webserverengine.h>* Abstract base class for a web server engine.

Such an engine implements an http server which a browser can connect to.

### Public Functions

**WebServerEngine** (bool *withMainModule*, int *minport*, int *maxport*, QString *externalUrl*,  
QString *dispatcherUrl*, QByteArray *wtoken*)

**~WebServerEngine** ()

QString **url** () = 0

Returns the home url of this engine (where the end user can point the browser to). .

QString **externalUrl** ()

Returns the external root url for creating full externally valid urls (even behind reverse proxies).

Returns *url()* if no external root url is specified.

Change this by the cmdline parameter '*externalUrl*'.

QString **dispatcherUrl** ()

Returns the internal root url of the dispatcher (workers only).



QByteArray **dispatcherWtoken** ()

Returns the wtoken got from the dispatcher (workers only).

QUrl **getFullCommandUrl** (QString *command*, [WebCommandData](#) *data*, bool *external*)

Generates a url path for a command.

Typically only needed for full url generation on server side.

void **triggerEvent** (QString *name*, QJsonObject *data*)

Triggers an event (at the clients).

void **triggerEvent** (QString *name*, QJsonValue *data* = QJsonValue::Undefined)

QString **getConfigValue** (QString *key*)

Returns the value for a runtime configuration key.

void **setConfigValue** (QString *key*, QString *value*)

Sets the runtime configuration key to a value (triggering a event updating the clients).

void **getStaticFile** ([WebServerEngineRequest](#) \**request*, QString *path*)

Returns a static file content (for the app itself) on a request.

QUrl **rebaseUrl** (QUrl *rootUrl*, QUrl *url*)

Returns a url rebased to another engine's root url.

Used for dispatching to workers.

void **addAndPreserveModule** (std::shared\_ptr<[sh::ui::web::WebModule](#)> *module*)

Plugs a [WebModule](#) into the engine and holds a pointer to it. Unplug it by calling [removeModule\(\)](#).

void **addModule** (std::shared\_ptr<[sh::ui::web::WebModule](#)> *module*)

Plugs a [WebModule](#) into the engine. Unplug it by either delete module (drop all pointers to it) or by calling [removeModule\(\)](#).

void **removeModule** ([sh::ui::web::WebModule](#) \**module*)

Unplugs a [WebModule](#) from the engine.

qint64 **currentTimestamp** ()

Returns the current engine timestamp.

It's not related to real time, but just a value which is higher each time you query it.

This is used for coordination of some time-order sensitive operations on browser side.

## Public Static Functions

[WebServerEngine](#) \***createDispatcherEngine** ()

Creates and returns a web engine for a dispatcher.

[WebServerEngine](#) \***createWorkerEngine** (QString *dispatcherUrl*, QByteArray *wtoken*)

Creates and returns a web engine for a worker.

### Parameters

- *dispatcherUrl*: The root url of the dispatcher.

### Private Members

```
const QJsonObject _jok
QMutex _configValuesMutex
QMutex _modulesMutex
int _minport
int _maxport
QUrl _externalurl
QUrl _dispatcherurl
QMap<QString, QString> _configValues
QList<std::weak_ptr<sh::ui::web::WebModule>> _modules
QList<std::shared_ptr<sh::ui::web::WebModule>> _smodules
std::shared_ptr<sh::ui::web::WebServerEngineMainModule> _mainmodule
QString _webstaticpath
qint64 _currentTimestamp = 1
QByteArray _wtoken
```

### Private Static Functions

```
WebServerEngine *_createEngine (bool withMainModule,   QString dispatcherUrl,
                                QByteArray wtoken)
```

### Friends

```
friend class WebServerEngineMainModule
```

```
class WebServerEngineDispatcher : public QObject, public sh::ui::web::WebModule
    #include <webserverenginedispatcher.h> The web serve engine dispatcher module, used in dispatcher
    mode for providing a portal url which starts Shallot instances on request and internally redirects to
    them.
```

### Public Functions

```
WebServerEngineDispatcher ()
```

```
~WebServerEngineDispatcher ()
```

```
void stop ()
    Stops the dispatcher.
```

```
void setEngine (WebServerEngine *webserver)
    Assigns this web module to a sh::ui::web::WebServerEngine. .
```

```
WebServerEngine *webserver ()
    Returns the sh::ui::web::WebServerEngine this web module is assigned to (may be nullptr).
```

## Public Static Functions

*WebMainWindow* \***handleDispatching** (std::function<*WebMainWindow*\*) QString dispatcherUrl, QByteArray wtoken  
 > *createWndFctHandles* dispatching and creates a new main window (in child processes) on demand via a given factory function.

void **doInitialize** ()

void **doShutdown** ()

bool **executeCommand\_byQObjectReflection** (QObject \*o, *WebServerEngineRequest* \*request, QString command, QString justLeftside = QString())

Convenience function often used by *executeCommand*()).

For a QObject, it searches there for a slot with the name cmd\_{command}, with ``'`'s replaced by '\_'s (so, for the command"foo/get\_bars", it searches for slotcmd\_foo\_get\_bars`). If it finds one, it gets executed (in a worker thread). The request is considered as handled, if so.

If there is such a slot, but with appended \_\_M in name, it is called in main thread!

**Return** If it handled (answered) the command.

## Private Functions

bool **compareHashRedirectorWebworkerUrl** (QByteArray hash, QString url, QString wtoken)  
 Compares a hash for a redirector webworker url to other data.

*WorkerProcess* \***spawnNewWorker** (*WebServerEngineRequest* \*request)  
 Spawns a new worker.

Does not execute any requests on it.

*WorkerProcess* \***findWorkerForWtoken** (QString wtoken)  
 Returns the worker for a wtoken.

QString **wtokenFromRequestCookie** (*WebServerEngineRequest* \*request)  
 Returns the wtoken from a request (typically from cookie).

bool **executeCommand** (*WebServerEngineRequest* \*request, QString command) **override**  
 Executes command as requested by the browser, if responsible.

Read parameters, check if this module is responsible for answering, and then write an answer with request.

Implementations often use *executeCommand\_byQObjectReflection*() for convenience.

**Return** If it handled (answered) the command.

bool **rootCommand** (*WebServerEngineRequest* \*request, QString webadapter) **override**  
 Returns the root ('index.html') content. .

### Private Members

```
std::shared_ptr<WebServerEngine> _webserver
QMutex _workersmutex
QHash<QString, WorkerProcess*> _workers
JanitorThread _janitor
```

### Private Slots

```
void cmd__drop_worker__M (WebServerEngineRequest *request)
void cmd__set_last_user_activity (WebServerEngineRequest *request)
void cmd__set_prevent_close_message (WebServerEngineRequest *request)
void cmd__working_for (WebServerEngineRequest *request)
```

### Private Static Functions

```
void start (QUrl &url)
    Starts the dispatcher engine, accepting http connections and internally handling child processes.
QByteArray hashRedirectorWebworkerUrl (QString url, QString wtoken)
    Compute a hash for a redirector webworker url.
```

### Private Static Attributes

```
std::shared_ptr<WebServerEngineDispatcher> _dispatcher = nullptr
QString _dispatcherId = QString::fromLatin1(sh::tools::Misc::generateUniqueHash())
QString _dispatcherCookieName = QUrl::toPercentEncoding(QString("shallot_%1").arg(_dispatcherId)).toLocal
class JanitorThread : public QThread
```

### Public Functions

```
JanitorThread (WebServerEngineDispatcher *dispatcher)
void run () override
```

### Private Functions

```
qint64 machineMemory ()
    Returns the amount of memory this machine provides (in bytes).
    Returns -1 if this is not supported on the target platform or it failed.
void stopWorker (WorkerProcess *worker, QString reason)
int detectGoodMaxNumberWorkersByMachineCapabilities ()
QList<WorkerProcess*> stopWorkers (QList<WorkerProcess*> workers, int stopcount,
    QString reason)
```

## Private Members

*WebServerEngineDispatcher* \*\_dispatcher

double \_allMemory

QHash<*WorkerProcess*\*, double> \_workerMemoryUsagesLastSeconds

**class WorkerProcess : public QProcess**

A internal Shallot worker process driving one Shallot session.

## Public Functions

QByteArray **wtoken** ()

The wtoken (used for association and security).

QUrl **rootUrl** ()

The worker root url.

void **request** (*WebServerEngineRequest* \*request)

Make a request to the worker.

qint64 **workerPid** ()

Returns the process id of the worker.

qint64 **memoryUsage** ()

Returns the current memory usage of this worker (in bytes).

Returns -1 if this is not supported on the target platform or it failed.

QDateTime **lastBrowserHeartbeat** ()

Returns when a browser was seen connected to this worker last time.

QDateTime **lastUserInteraction** ()

Returns when a user has interacted with this worker last time.

QString **clientHost** ()

Returns the client host this worker is serving.

It's not guaranteed to have some specific content (maybe an ip address), but is equal for the same host, some one can count how many hosts work for one particular client host.

QString **preventCloseMessage** ()

Returns a reason message why this worker currently should not be closed (or "" if closing is fine).

See also `isClosingInappropriate()`.

Note: This is reported asynchronously. You must not rely on its precise information, but solely use it for monitoring.

bool **isClosingFine** ()

Returns if it is fine to stop this worker (or if there is a good reason to not do).

See also `preventCloseMessage()`.

Note: This is reported asynchronously. You must not rely on its precise information, but solely use it for monitoring.

## Private Functions

**WorkerProcess** (*WebServerEngineDispatcher* \*dispatcher, QByteArray wtoken, QString clientHost, QStringList acceptedLanguages)

bool **waitOnline** ()

Waits until the worker is online and ready for requests (or dead).

void **initialize** (QString rooturl, qint64 pid)

Initializes the worker instance (which exists on the dispatcher side!) by setting some final data.

void **gotBrowserHeartbeat** ()

Refreshes the browser heartbeat timestamp.

See *lastBrowserHeartbeat()*.

void **gotUserInteraction** (int value)

Refreshes the user interaction timestamp.

See *lastUserInteraction()*.

void **setPreventCloseMessage** (QString message)

Sets the preventCloseMessage (empty: closeable).

## Private Members

*WebServerEngineDispatcher* \*\_dispatcher

QByteArray \_wtoken

QUrl \_rooturl

QMutex \_mutex

QWaitCondition \_cnd\_initied

qint64 \_pid = -1

QDateTime \_lastBrowserHeartbeat

QDateTime \_lastUserInteraction

QString \_clientHost

QString \_preventCloseMessage

QRegularExpression \_reProcStatus

## Friends

**friend class** WebServerEngineDispatcher

**class** **WebServerEngineMainModule** : **public** QObject, **public** *sh::ui::web::WebModule*  
#include <webserverengine.h> The web serve engine main module, providing some base services like events.

## Public Functions

**WebServerEngineMainModule** ()

**~WebServerEngineMainModule** ()

bool **executeCommand** (*WebServerEngineRequest* \*request, QString command) **override**

Executes command as requested by the browser, if responsible.

Read parameters, check if this module is responsible for answering, and then write an answer with request.

Implementations often use *executeCommand\_byQObjectReflection()* for convenience.

**Return** If it handled (answered) the command.

bool **rootCommand** (*WebServerEngineRequest* \*request, QString webadapter) **override**

Returns the root ('index.html') content. .

void **triggerEvent** (QString name, QString data)

Triggers an event on browser side.

void **setEngine** (*WebServerEngine* \*webserver)

Assigns this web module to a *sh::ui::web::WebServerEngine*. .

*WebServerEngine* \*webserver ()

Returns the *sh::ui::web::WebServerEngine* this web module is assigned to (may be nullptr).

## Public Static Functions

bool **executeCommand\_byQObjectReflection** (QObject \*o, *WebServerEngineRequest* \*request, QString command, QString justLeftside = QString())

Convenience function often used by *executeCommand()*.

For a QObject, it searches there for a slot with the name cmd\_{command}, with ``'s replaced by '\_'s (so, for the command "foo/getBars", it searches for slotcmd\_foo\_getBars). If it finds one, it gets executed (in a worker thread). The request is considered as handled, if so.

If there is such a slot, but with appended \_\_M in name, it is called in main thread!

**Return** If it handled (answered) the command.

## Private Functions

qint64 **lastRequestTime** ()

## Private Members

qint64 **\_lastRequestTime**

QMutex **\_mutex\_lastRequestTime**

QList<EventEntry\*> **\_eventEntries**

quint64 **\_eventEntriesLastId** = 0

QMutex **\_mutex\_eventEntries**

QWaitCondition **\_eventEntriesCondition**

*EventEntryCleanupThread* **\_eventEntryCleanupThread**

*StopWhenIdleThread* **\_stopWhenIdleThread**

### Private Slots

void **cmd\_dispatcher\_toclient\_heartbeat** (*WebServerEngineRequest* \*request)

void **cmd\_events\_get\_next** (*WebServerEngineRequest* \*request)

### Private Static Attributes

QRegularExpression **\_restatic**

**class EventEntry**

Data for one occurred event.

### Public Functions

**EventEntry** (qint64 *id*, qint64 *time*, QString *name*, QString *data*)

### Public Members

const qint64 **id**

const qint64 **time**

const QString **name**

const QString **data**

### Friends

**friend class** EventEntryCleanupThread

**class EventEntryCleanupThread**: **public** QThread

Thread for cleaning up old *EventEntry* instances.

### Public Functions

**EventEntryCleanupThread** (*WebServerEngineMainModule* \*module)

### Private Members

*WebServerEngineMainModule* \***module**

**class StopWhenIdleThread**: **public** QThread

Thread for stopping the application if the remote side is gone.



## Public Functions

**StopWhenIdleThread** (*WebServerEngineMainModule* \*module)

## Private Members

*WebServerEngineMainModule* \*module

**class WebServerEngineRequest**

#include <webserverengine.h> A web request in a *WebServerEngine*.

Override this together with *WebServerEngine*.

## Public Functions

**WebServerEngineRequest** (QString url, QString clientHost)

QUrl **url** ()

Returns the requested url.

Note: The url should be considered as opaque, since the exact structure depends on the engine implementation.

QString **clientHost** ()

Returns the client host address.

QString **path** ()

Returns the path part of the requested url.

Note: The url should be considered as opaque, since the exact structure depends on the engine implementation.

QString **data** (QString key)

Returns data passed as parameter along with the request by key.

int **responseCode** ()

Returns the answered (http) response code.

QString **responseMimeType** ()

Returns the mimetype of the answer content.

QByteArray **response** ()

Returns the answer content.

bool **responseSet** ()

Returns if a response was set.

void **setResponse** (QByteArray response, QString mimetype, int responseCode = *HTTP\_OK*)

Sets the response.

void **setResponse** (QJsonArray response, int responseCode = *HTTP\_OK*)

void **setResponse** (QJsonObject response, int responseCode = *HTTP\_OK*)

QString **headerData** (QString key) = 0

Returns an http header value by key. .

QString **getCookie** (QString key) = 0

Returns an http cookie stored on browser side.

Note: This only works on dispatcher level and not in usual *WebModule* commands.

void **setCookie** (QString *key*, QString *value*, int *maxAgeSecs* = -1) = 0  
Sets an http cookie to be returned to the browser side.

Note: This only works on dispatcher level and not in usual *WebModule* commands.

### Public Static Attributes

```
const int HTTP_OK = 200
const int HTTP_SEE_OTHER = 303
const int HTTP_NOT_FOUND = 404
const int HTTP_INTERNAL_SERVER_ERROR = 500
const int HTTP_BAD_GATEWAY = 502
```

### Private Members

```
QUrl _url
QString _clientHost
QString _path
QMap<QString, QString> _getdata
int _responseCode = HTTP_NOT_FOUND
QString _responseMimeType = "application/octet-stream"
QByteArray _response
bool _responseSet = false
```

```
class WebStoreProfileDialog : public QObject, public sh::ui::web::WebDialog, public sh::ui::StoreProfileDialog
#include <webstoreprofiledialog.h> Web based save settings dialog.
```

### Public Functions

```
WebStoreProfileDialog (std::shared_ptr<const sh::filesystem::Eurl> eurl)
```

```
QList<SettingEntry> checkedSettings () override
Returns the list of settings the user has checked to store.
```

```
QString profileName () override
Returns the profile the user has chosen to store settings for.
```

```
bool withSubDirectories () override
Returns if the user has chosen to apply the checked settings also for subdirectories.
```

```
QStringList inheritsFrom () override
Returns the list of profiles the user has chosen to inherit from.
```

```
bool hasGlobal () override
Returns if the user has chosen to apply the checked settings for each directory (instead of the current directory).
```

```
QVariant dialogProperty (QString dlgPropertyKey, QVariant deflt = QVariant())
Returns a dialog property value by key.
```

void **setDialogProperty** (QString *dlgPropertyKey*, QVariant *value*)

Sets a dialog property value for a key.

QVariantHash **dialogResult** ()

Returns the dialog result as hash.

The browser side of a web dialog can ‘accept’ a dialog while closing by setting a non-empty dialog result. This mechanism allows browser side dialogs to return data to the backend. It’s up to the backend to interpret the content of this hash.

If the dialog was cancelled, or is still open, the returned hash is empty. See also `wasClosed()`.

void **triggerDialogEvent** (QString *name*, QJsonObject *data* = QJsonObject())

Triggers a dialog event (at the browsers).

They in turn typically fetch some data again in order to refresh the ui.

void **setCloseableByUser** (bool *v* = true)

Sets if this dialog shall be directly closeable by the user by the window decoration. Set to `false` if the user has to react on this dialog by clicking on some controls in the dialog body.

This method must be called during construction.

bool **isCloseableByUser** ()

Returns if this dialog shall be directly closeable by the user by the window decoration.

QJsonObject **toJson** () **override**

Returns the json representation as QJsonObject.

std::shared\_ptr<const *sh::filesystem::Eurl*> **eurl** ()

Returns the current directory this dialog shall work on.

qint64 **dialogId** ()

Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

bool **isInitiated** ()

Returns if this dialog is initialized.

Must be called in main thread.

bool **wasAccepted** ()

Returns if this dialog was ‘accepted’ by the user (typically by clicking some ‘OK’ button).

Must be called in main thread.

void **waitClosed** ()

Wait until the user closed the dialog in some way.

May be called in any thread.

void **close** ()

Closes the dialog.

Must be called in main thread.

bool **wasClosed** ()

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \***manager** ()

Returns the *DialogManager* which hosts this dialog.

## Private Slots

void **cmd\_\_get\_\_M** (*WebServerEngineRequest \*request*)

void **cmd\_\_store\_mode\_\_M** (*WebServerEngineRequest \*request*)

**class WebTuningDialog**: public QObject, public *sh::ui::web::WebDialog*, public *sh::ui::TuningDialog*  
#include <webtuningdialog.h> Web based tuning dialog.

## Public Functions

**WebTuningDialog** ()

QVariant **dialogProperty** (QString *dlgPropertyKey*, QVariant *deft* = QVariant())  
Returns a dialog property value by key.

void **setDialogProperty** (QString *dlgPropertyKey*, QVariant *value*)  
Sets a dialog property value for a key.

QVariantHash **dialogResult** ()  
Returns the dialog result as hash.

The browser side of a web dialog can ‘accept’ a dialog while closing by setting a non-empty dialog result. This mechanism allows browser side dialogs to return data to the backend. It’s up to the backend to interpret the content of this hash.

If the dialog was cancelled, or is still open, the returned hash is empty. See also wasClosed().

void **triggerDialogEvent** (QString *name*, QJsonObject *data* = QJsonObject())  
Triggers a dialog event (at the browsers).

They in turn typically fetch some data again in order to refresh the ui.

void **setCloseableByUser** (bool *v* = true)  
Sets if this dialog shall be directly closeable by the user by the window decoration. Set to *false* if the user has to react on this dialog by clicking on some controls in the dialog body.

This method must be called during construction.

bool **isCloseableByUser** ()  
Returns if this dialog shall be directly closeable by the user by the window decoration.

QJsonObject **toJson** () **override**  
Returns the json representation as QJsonObject.

qint64 **dialogId** ()  
Returns the dialog id.  
  
Each instance has an id unique in the complete Shallot process lifetime.  
  
Must be called in main thread.

bool **isInited** ()  
Returns if this dialog is initialized.  
  
Must be called in main thread.

bool **wasAccepted** ()  
Returns if this dialog was ‘accepted’ by the user (typically by clicking some ‘OK’ button).  
  
Must be called in main thread.

void **waitClosed**()  
 Wait until the user closed the dialog in some way.  
 May be called in any thread.

void **close**()  
 Closes the dialog.  
 Must be called in main thread.

bool **wasClosed**()  
 Returns if this dialog was closed.  
 Must be called in main thread.

*DialogManager* \***manager**()  
 Returns the *DialogManager* which hosts this dialog.

### Private Members

QMutex **\_cfgvaluesmutex**  
 QWaitCondition **\_cfgvaluescondition**  
 bool **\_cfgvaluesinitd** = false  
 QList<std::shared\_ptr<sh::configuration::ConfigurationValue>> **\_cfgvalues**

### Private Slots

void **cmd\_\_change\_configurationvalue** (*WebServerEngineRequest* \*request)  
 void **cmd\_\_get\_configurationvalues** (*WebServerEngineRequest* \*request)

### Private Static Functions

QString **categoryToName** (sh::configuration::ConfigurationCategory category)  
 QString **valueTypeToName** (sh::configuration::ConfigurationValueType \*valueType)  
 QObject **configurationValueToJsonObject** (std::shared\_ptr<sh::configuration::ConfigurationValue> cfval)

**namespace engines**  
 Web server low-level engine implementations.

## 10.2.26 Namespace sh::ui::qt

**namespace sh::ui::qt**  
 The Qt user interface implementation.

This is the default UI. It builds a desktop application based on the Qt gui components.

**class LineEditWithKeyboardShortcuts** : public QLineEdit  
*#include <qtjumpbar.h>* Helper widget of *QtJumpBar*.

## Public Functions

**LineEditWithKeyboardShortcuts** (QWidget \*parent = 0)

## Signals

void **enterPressed** ()

void **escapePressed** ()

**class** **QtAboutDialog**: public *sh::ui::qt::QtDialog*, public *sh::ui::AboutDialog*  
*#include <qtaboutdialog.h>* Qt based about dialog.

## Public Functions

**QtAboutDialog** ()

**~QtAboutDialog** ()

qint64 **dialogId** ()

Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

bool **isInitied** ()

Returns if this dialog is initialized.

Must be called in main thread.

bool **wasAccepted** ()

Returns if this dialog was ‘accepted’ by the user (typically by clicking some ‘OK’ button).

Must be called in main thread.

void **waitClosed** ()

Wait until the user closed the dialog in some way.

May be called in any thread.

void **close** ()

Closes the dialog.

Must be called in main thread.

bool **wasClosed** ()

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \***manager** ()

Returns the *DialogManager* which hosts this dialog.

## Private Members

*Ui*::QtAboutDialog \*ui

## Private Slots

void on\_btnClose\_clicked()

void on\_btnLicense\_clicked()

void on\_btnHomepage\_clicked()

**class** QtActionExecutionInfoDialog : public QDialog, public *sh::ui::ActionExecutionInfoDialog*  
*#include <qtactionexecutioninfodialog.h>* Qt progress dialog for action executions.

## Public Types

**enum** MessageBoxButton

Buttons in a message box from *ActionExecutionUserFeedback*.

*Values:*

**enumerator** NONE = 0

**enumerator** OK = 1 << 0

**enumerator** Continue = 1 << 1

**enumerator** Cancel = 1 << 2

**enumerator** Retry = 1 << 3

**enumerator** Yes = 1 << 4

**enumerator** No = 1 << 5

## Public Functions

**QtActionExecutionInfoDialog** (*sh::actions::ActionExecutionInfo* \*info, QWidget \*parent = 0)

**~QtActionExecutionInfoDialog** ()

void **setDetails** (QString fv, QString fob, QString tv, QString tob)

Sets the item details text ('from a/foo.jpg', 'to b/foo.jpg'). .

void **setHead** (QString txt)

Sets the header text. .

void **setProgress** (bool isDeterminate, quint64 done, quint64 all, QString text)

Sets the progress. .

int **messageBox** (QString text, QList<QString> answers, QString icon = QString(), int defaultanswer = -1, int cancelanswer = -1, QList<QString> answericons = QList<QString>())

int **inputBox** (QString text, QList<QString> answers, QString \*value, QString icon = QString(), int defaultanswer = -1, int cancelanswer = -1, int valuePreselectFrom = -1, int valuePreselectTo = -1)

int **multilineInputBox** (QString text, QList<QString> answers, QString \*value, QString icon = QString(), int defaultanswer = -1, int cancelanswer = -1)

```
int simpleChooserGridform (QString text, GridformEntries *entries, QStringList answers, int
                           defaultanswer = -1, int cancelanswer = -1)

bool credentialsDialog (QString text, bool showDomain, bool showUsername, bool show-
                        Password, bool showAnonymous, bool showRemember, QString *do-
                        main, QString *username, QString *password, bool *isAnonymous,
                        bool *isRemember)

bool unixPermissionsDialog (bool *userMayRead, bool *userMayWrite, bool *userMayEx-
                            ecute, bool *groupMayRead, bool *groupMayWrite, bool
                            *groupMayExecute, bool *othersMayRead, bool *othersMay-
                            Write, bool *othersMayExecute, bool *sticky, bool *setuid,
                            bool *setgid, QStringList users, QStringList groups, QString
                            *ownerUser, QString *ownerGroup)

void setLogicallyVisible (bool v)
    Sets if this dialog is logically visible (i.e. set visible by the action). .

void setBackground (bool v)
    Sets if this dialog is currently in background mode (i.e. not visible). .

void setForceForeground (bool v)
    Sets if this dialog is currently forced to be visible in foreground. .

bool isLogicallyVisible ()
    Returns if this dialog is logically visible (i.e. set visible by the action).

bool isBackground ()
    Returns if this dialog is currently in background mode (i.e. not visible).

QString simpleInputDialog (QString text, QString deft, int valuePreselectFrom = -1, int valuePre-
                           selectTo = -1)

int simpleMessageBox (QString text, int buttons = (int)MessageBoxButton::OK, QString icon =
                     QString(), int defaultbutton = (MessageBoxButton)0, int cancelbutton =
                     (MessageBoxButton)0)
```

## Private Functions

```
void _computevisibility ()
```

## Private Members

```
Ui::QtActionExecutionInfoDialog *ui
```

```
sh::ui::qt::feedbackpanels::FeedbackPanel *currentFeedbackPanel = 0
```

## Private Slots

```
void on_btnCancel_clicked ()
```

```
void on_btnBackground_clicked ()
```

```
class QtActionExecutionInfoPanel : public QWidget, public sh::ui::ActionExecutionInfoPanel
    #include <qtactionexecutioninfopanel.h> Qt status bar info-panel for an action execution.
```



## Public Functions

```

QtActionExecutionInfoPanel (sh::actions::ActionExecutionInfo *info, QColor color,
                             QWidget *parent = 0)

void setLabel (QString s)
    Sets the label text. .

void setWidth (int width)
    Sets the panel with (in pixels). .

QSize sizeHint () const

void setProgress (bool isProgressDeterminate, quint64 progressDone, quint64 progressAll)
    Sets the progress. .

void setForceForeground (bool v)
    Sets if the associated action is currently forced to be visible in foreground. .

void setPanelVisible (bool v)
    Sets if the panel is visible. .

void onClicked (std::function<void>)
    > fcnQObject *owner = 0 Sets a handler for a click on the button (optionally bound to an owner life-
    time).

void onDestroyed (std::function<void>)
    > fcnQObject *owner = 0 Sets a handler for panel removal (optionally bound to an owner lifetime).

void onVisibilityChanged (std::function<void>)
    > fcnQObject *owner = 0 Sets a handler for panel visibility changes (optionally bound to an owner
    lifetime).

```

## Private Functions

```

void _check_indeterminateanimationtimer_enable ()

```

## Private Members

```

QString lb11

bool _progressDeterminate = false

int _indeterminateAnimationCounter = 0

int _width

QTimer * _indeterminateAnimationTimer

quint64 _progressAll

quint64 _progressDone

bool _isforeground_forced = false

sh::actions::ActionExecutionInfo * info

QPixmap gradient

QColor colorProgress2

QColor colorProgress1

```

```
    QColor colorBase
    QColor colorFrame
    QColor colorText
    QColor colorTextDark

class QtActionMenu : public QMenu
    #include <qtactionmenu.h> Qt based action menu used for context menus and in the toolbar.
```

### Public Functions

```
QtActionMenu (QWidget *parent = 0)
void setHeader (QString t)
QList<QAction*> allActions ()
void clearAllActions ()
void removeActionAt (int i)
QAction *addNewAction (int i = -1)
QAction *addNewSubMenu (sh::ui::qt::QtActionMenu **qsubmenu, int i = -1)
QAction *addNewSeparator (int i = -1)
QAction *addNewHeader (int i = -1)
bool actionIsHeader (QAction *a)
```

### Private Functions

```
void _observeAction (QAction *a)
void _correctMenuPosition ()
```

### Private Members

```
    QColor brandingcolor
    QFont _boldfont
    QFont _normalfont
    int _origMenuPosX = -1
    int _origMenuPosY = -1
    QString _header
    int _cntInternalActions

class QtActionMenuHandler
    #include <qtactionmenuhandler.h> A handler which reflects the sh::actions objects to a graphical menu
    (and keeps that up-to-date).
```

## Public Functions

**QtActionMenuHandler** (std::shared\_ptr<sh::actions::ActionInstantiation> *ai*,  
std::shared\_ptr<QtActionMenu> *menu*)

## Private Functions

void **\_markDefault2** ()

void **\_append\_actions** (sh::ui::qt::QtActionMenu \**menu*, QList<std::shared\_ptr<sh::actions::ActionInstantiation>>  
*acts*, std::shared\_ptr<sh::actions::ActionCategory> *category*)

## Private Members

QList<std::shared\_ptr<sh::actions::ActionInstantiation>> **selects**

QList<std::shared\_ptr<sh::actions::ActionInstantiation>> **diracts**

QHash<QAction\*, std::shared\_ptr<sh::actions::AbstractActionItem>> **qaction2action**

std::weak\_ptr<QtActionMenu> **menu**

## Private Static Functions

std::shared\_ptr<sh::actions::ActionActionItem> **\_getDefaultAction** (QList<std::shared\_ptr<sh::actions::AbstractActionItem>>  
*actionList*)

void **\_markDefault** (std::weak\_ptr<sh::actions::SubmenuItem> *itmSubmenu*,  
sh::ui::qt::QtActionMenu \**menu*)

QAction \* **\_createAndConnectAction** (sh::actions::AbstractActionItem \**itm*,  
sh::ui::qt::QtActionMenu \**menu*, std::function<void>  
> *onchanged*QObject \**onchangedbuddy* = 0

void **\_applyPropertiesToQAction** (sh::actions::AbstractActionItem \**itm*, QAction \**\_wid-*  
*getaction*)

void **\_updateSubmenu** (sh::actions::SubmenuItem \**itm*, QtActionMenu \**menu*,  
std::function<void>  
> *onchanged* = 0

## Friends

**friend class** QtToolBarButtonHandler

**class QtDialog**: public QDialog

#include <qtdialog.h> Abstract base class for a qt based dialog. Typically used for also implementing some sh::ui::Dialog.

Subclassed by sh::ui::qt::QtAboutDialog, sh::ui::qt::QtExceptionDialog,  
sh::ui::qt::QtFilePropertyDialog, sh::ui::qt::QtLogViewDialog, sh::ui::qt::QtManageBookmarksDialog,  
sh::ui::qt::QtManageProfilesDialog, sh::ui::qt::QtOpenWithDialog, sh::ui::qt::QtStoreProfileDialog,  
sh::ui::qt::QtTuningDialog

## Public Functions

**QtDialog()**

**class QtDialogManager** : public *sh::ui::DialogManager*  
*#include <qtdialog.h>* A *DialogManager* used in Qt ui.

## Public Functions

**std::shared\_ptr<Dialog> getDialogById** (qint64 *id*)

Returns a *Dialog* by dialog id.

Must be called in main thread.

**QList<std::shared\_ptr<Dialog>> getAllDialogs** ()

Returns a list of all dialogs currently shown (in order of creation).

Must be called in main thread.

**QList<qint64> getAllDialogIds** ()

Returns a list of dialog ids of all dialogs currently shown (in order of creation).

Must be called in main thread.

**template<class TDlg, typename ...Args>**

**std::shared\_ptr<TDlg> createAndShowDialog** (*Args... args*)

Creates and shows a *Dialog* by class and constructor parameters.

Those dialogs (TDlg) have to implement *Dialog* (typically a subclass of it, representing a particular dialog, like *FilePropertyDialog*), and also some ui mode (e.g. qt, web) specific class (depends on that ui mode).

You typically should not have to use it outside of *MainWindow* or subclasses. The *MainWindow* interface allows to create all available dialogs in a ui mode independent way.

May be called in any thread.

**Return** The created *Dialog* instance.

## Private Functions

**void showDialog** (std::shared\_ptr<Dialog> *dialog*) **override**

This method implements the ui mode specific mechanism for showing a dialog.

Must be called in main thread.

**class QtExceptionDialog** : public *sh::ui::qt::QtDialog*, public *sh::ui::ExceptionDialog*  
*#include <qexceptiondialog.h>* Qt based exception dialog.

## Public Functions

**QtExceptionDialog** (QString *error1*, QString *error2*, QString *details*, QString *icon*, bool *mayRetry*, bool *mayClose*, bool *mayCancel*, bool *showLoglabelAndDetails*)

**~QtExceptionDialog** ()

*ExceptionDialogResult* **answer** () **override**

Returns the answer the user has given in this dialog.

QString **error1** ()

Returns the 1st error text.

QString **error2** ()

Returns the 2nd error text.

QString **details** ()

Returns the error details.

QString **icon** ()

Returns the icon (as name resolveable by *sh::base::IconManager*).

bool **mayRetry** ()

Returns if it's allowed to retry.

bool **mayClose** ()

Returns if it's allowed to just close and continue without closing Shallot.

bool **mayCancel** ()

Returns if it's allowed to cancel, i.e. throwing a *sh::exceptions::CancelException*.

bool **showLoglabelAndDetails** ()

Returns if the dialog shall allow to read the details.

qint64 **dialogId** ()

Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

bool **isInitiated** ()

Returns if this dialog is initialized.

Must be called in main thread.

bool **wasAccepted** ()

Returns if this dialog was 'accepted' by the user (typically by clicking some 'OK' button).

Must be called in main thread.

void **waitClosed** ()

Wait until the user closed the dialog in some way.

May be called in any thread.

void **close** ()

Closes the dialog.

Must be called in main thread.

bool **wasClosed** ()

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \*manager ()

Returns the *DialogManager* which hosts this dialog.

### Private Members

*Ui::QtExceptionDialog* \*ui

*ExceptionDialogResult* \_answer = *ExceptionDialogResult::Shutdown*

### Private Slots

void on\_btnExit\_clicked ()

void on\_btnClose\_clicked ()

void on\_btnCancel\_clicked ()

void on\_btnRetry\_clicked ()

void on\_btnShowLog\_clicked ()

void on\_btnExitAndSaveLog\_clicked ()

void on\_btnShowDetails\_toggled (bool checked)

**class** QtFileDetailsPanel : public QWidget

*#include <qtfiledetailspanel.h>* The details panel.

Can be shown in the main window. It presents detail information about the selected file.

### Public Functions

**QtFileDetailsPanel** (QWidget \*parent = 0)

void **setNodes** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> nodes)

void **setOrientation** (Qt::Orientation orientation)

QSize **sizeHint** () **const override**

**~QtFileDetailsPanel** ()

### Private Members

int PADDINGX

int PADDINGY

QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> \_nodes

QList<std::shared\_ptr<*sh::paneldetails::PanelDetail*>> \_panelDetails

QList<*DetailPlacement*> \_placements

QFont fontNormal

QFont fontBold

QPixmap \_widgetimagecache

QTimer \_widgetimagecachetimer

Qt::Orientation **\_orientation** = Qt::Horizontal

```
class DetailPlacement
    #include <qtfiledetailspanel.h>
```

### Public Functions

**DetailPlacement** (int *x*, int *y*, int *w*, int *h*, QList<[DetailRowPlacement](#)> *rowplacements*,  
std::shared\_ptr<[sh::paneldetails::PanelDetail](#)> *detail*)

### Public Members

std::shared\_ptr<[sh::paneldetails::PanelDetail](#)> **detail**

int **x**

int **y**

int **w**

int **h**

QList<[DetailRowPlacement](#)> **rowplacements**

```
class DetailRowPlacement
    #include <qtfiledetailspanel.h>
```

### Public Functions

**DetailRowPlacement** (int *h*, int *contentx*, int *contenty*, QList<int> *elementwidths*)

**DetailRowPlacement** () = default

**DetailRowPlacement** (const [DetailRowPlacement](#)&) = default

### Public Members

int **h**

int **contentx**

int **contenty**

QList<int> **elementwidths**

```
class QtFileIconview : public QListView, public sh::ui::qt::QtFileView
    #include <qtfileiconview.h> Icon view for the contents of one directory.
```

## Public Functions

**QtFileIconview** (QWidget *\*parent* = 0)  
Constructed only by the infrastructure and made available otherwise.

void **setThumbDimension** (double *size*)  
Sets the size of the thumbnail image.

void **setBackgroundColor** (QString *c*)

QString **backgroundColor** ()

QList<std::shared\_ptr<sh::filesystem::FilesystemNode>> **selectedNodes** ()

void **setSelection** (const QList<std::shared\_ptr<sh::filesystem::FilesystemNode>> *nodes*)

std::shared\_ptr<sh::filesystem::FilesystemNode> **node** ()

QList<std::shared\_ptr<sh::filesystem::FilesystemNode>> **getAllVisibleNodes** ()

void **setSizeFormatting** (sh::filesystem::SizeFormatting *v*)

void **gotoDir** (std::shared\_ptr<sh::filesystem::FilesystemNode> *n*)

void **setHiddenFilesVisible** (bool *value*)

QThread **\*thread** ()

void **configure** (sh::ui::qt::QtFileViewControl *\*viewctrl*)

void **focus** ()

void **setSort** (int *column*, Qt::SortOrder *order*)

QObject **\*as\_qobject** ()

QWidget **\*as\_qwidget** ()

QAbstractItemView **\*as\_qabstractitemview** ()

sh::ui::qt::QtFileViewControl **\*control** ()

## Private Members

int **\_deleg\_w** = 0

int **\_deleg\_h** = 0

int **\_dsx** = 0

int **\_dsy** = 0

int **\_lineheight** = 0

**class QtFileList** : public QTreeView, public sh::ui::qt::QtFileView  
#include <qtfilelist.h> List view for the contents of one directory.



## Public Functions

```

QtFileList (QWidget *parent = 0)
    Constructed only by the infrastructure and made available otherwise.

void setSort (int column, Qt::SortOrder order) override

void setBackgroundColor (QString c)

QString backgroundColor ()

QList<std::shared_ptr<sh::filesystem::FilesystemNode>> selectedNodes ()

void setSelection (const QList<std::shared_ptr<sh::filesystem::FilesystemNode>> nodes)

std::shared_ptr<sh::filesystem::FilesystemNode> node ()

QList<std::shared_ptr<sh::filesystem::FilesystemNode>> getAllVisibleNodes ()

void setSizeFormatting (sh::filesystem::SizeFormatting v)

void gotoDir (std::shared_ptr<sh::filesystem::FilesystemNode> n)

void setHiddenFilesVisible (bool value)

QThread *thread ()

void configure (sh::ui::qt::QtFileViewControl *viewctrl)

void focus ()

QObject *as_qobject ()

QWidget *as_qwidget ()

QAbstractItemView *as_qabstractitemview ()

sh::ui::qt::QtFileViewControl *control ()

```

## Private Functions

```

QString getColumnnameForIndex (int index)

void _adaptColumnWidth (int index)

```

## Private Members

```

bool _skip_slot_sectionResized = false

```

## Private Slots

```

void slot_sectionResized (int index, int oldsize, int newsize)

```

## Private Static Attributes

```
std::shared_ptr<sh::configuration::ConfigurationValue> cfgvalFileListIconSize = sh::configuration::ConfigurationValue::FileListIconSize;

class QtFilePropertyDialog : public sh::ui::qt::QtDialog, public sh::ui::FilePropertyDialog
    #include <qfilepropertydialog.h> Qt based properties dialog.
```

## Public Functions

**~QtFilePropertyDialog()**

**QtFilePropertyDialog**(QList<std::shared\_ptr<sh::filesystem::FilesystemNode>> nodes)  
Constructed only by the infrastructure and made available otherwise.

void **init**(std::shared\_ptr<Dialog> dialog) **override**  
Executes custom stuff on initialization, e.g. for populating the dialog.

void **refresh**() **override**  
Refreshes the dialog content.

sh::ui::FilePropertyDialogTabTableView \***createTabViewTable**() **override**  
Creates a new tab view table sub-widget.

sh::ui::FilePropertyDialogTabTextView \***createTabViewText**() **override**  
Creates a new tab view text sub-widget.

sh::ui::FilePropertyDialogTabIconTextBannerView \***createTabViewIconTextBanner**() **override**  
Creates a new tab view icon text banner sub-widget.

QList<std::shared\_ptr<FilePropertyDialogTab>> **tabs**()  
Returns a list of all tabs.

sh::ui::FilePropertyDialogTabActionsView \***widgetAt**(std::shared\_ptr<FilePropertyDialogTab> tab, int i)  
Returns the widget from a tab at a certain index.

int **widgetCount**(std::shared\_ptr<FilePropertyDialogTab> tab)  
Returns the number of widgets in a tab.

qint64 **dialogId**()  
Returns the dialog id.  
  
Each instance has an id unique in the complete Shallot process lifetime.  
  
Must be called in main thread.

bool **isInited**()  
Returns if this dialog is initialized.  
  
Must be called in main thread.

bool **wasAccepted**()  
Returns if this dialog was 'accepted' by the user (typically by clicking some 'OK' button).  
  
Must be called in main thread.

void **waitClosed**()  
Wait until the user closed the dialog in some way.  
  
May be called in any thread.

void **close** ()  
 Closes the dialog.  
 Must be called in main thread.

bool **wasClosed** ()  
 Returns if this dialog was closed.  
 Must be called in main thread.

*DialogManager* \***manager** ()  
 Returns the *DialogManager* which hosts this dialog.

## Public Slots

void **on\_btnClose\_clicked** ()  
 void **on\_btnRefresh\_clicked** ()  
 void **on\_tabWidget\_currentChanged** (int *index*)  
 void **selectTabByScrollPosition** ()

## Public Static Functions

void **addTabFactory** (int *i*, std::shared\_ptr<*FilePropertyDialogTabFactory*> *factory*)  
 Adds a *FilePropertyDialogTabFactory* so the Properties dialogs show its content.

See also the REGISTER\_FILEPROPERTYDIALOGTAB macro.

### Parameters

- *i*: An index which controls the display order. Use one of the REGISTER\_FILEPROPERTYDIALOGTAB\_INDEX\_\* values in *FilePropertyDialogTabFactory* as base values.

## Private Members

*Ui*::QtFilePropertyDialog \***ui**  
 QList<QWidget\*> **\_internaltabwidgets**  
 QList<QString> **\_tabheads**  
 int **\_skip\_on\_tabWidget\_currentChanged** = 0  
 int **\_skip\_selectTabByScrollPosition** = 0  
 QVBoxLayout \***scrollAreaLayout**

## Friends

```
friend class FilePropertyDialogTab  
class QtFilePropertyDialogTabActionsView : public QWidget, public sh::ui::FilePropertyDialogTabActionsView  
    #include <qtfilepropertydialogtabactionsview.h> A tab view which shows a main widget in the main part  
    and some buttons below.
```

## Public Functions

```
QtFilePropertyDialogTabActionsView (QWidget *parent = 0)  
    Is intended to be directly constructed from everywhere.  
  
void setContent (FilePropertyDialogTabViewContent *w)  
    Sets the main widget. .  
  
void setButtons (QList<QString> buttons)  
    Sets the list of buttons. .  
  
void setVisible (bool v)  
    Sets the view visible or hidden. .  
  
FilePropertyDialogTabViewContent *content ()  
    The main widget.  
  
QList<QString> buttons ()  
    The list of buttons.  
  
void onButtonTriggered (std::function<void>) int i  
    > fcn, QObject *owner = 0 Sets a handler for a click on one of the buttons (optionally bound to an  
    owner lifetime).  
  
class QtFilePropertyDialogTabIconTextBannerView : public QWidget, public sh::ui::FilePropertyDialogTabIconTextBannerView  
    #include <qtfilepropertydialogtabicontextbannerview.h> Qt based FilePropertyDialogTabIconTextBannerView.
```

## Public Functions

```
QtFilePropertyDialogTabIconTextBannerView (QWidget *parent = 0)  
  
void clear ()  
    Clears the contents.  
  
void insertIcon (QIcon icon, int sizePt = 24)  
    Adds an icon.  
  
void insertText (QString text)  
    Adds a text.
```

## Private Members

QHBoxLayout \*\_layout

**class QtFilePropertyDialogTabTableView** : public QTableView, public *sh::ui::FilePropertyDialogTabTableView*  
*#include <qtfilepropertydialogtabtableview.h>* Qt based *FilePropertyDialogTabTableView*.

## Public Types

**typedef** QPair<int, int> **ItemIndex**

## Public Functions

**QtFilePropertyDialogTabTableView** (QWidget \*parent = 0)

void **setContent** (QList<QPair<QString, QString>> content)  
 Sets the content.

QList<*ItemIndex*> **getSelectedItems** ()  
 Returns the selected cells.

QVariant **getItemValue** (int r, int c)  
 Returns a value of a cell.

## Private Functions

void **createAndSetModel** ()

**class QtFilePropertyDialogTabTextView** : public QLabel, public *sh::ui::FilePropertyDialogTabTextView*  
*#include <qtfilepropertydialogtabtextview.h>* Qt based *FilePropertyDialogTabTextView*.

## Public Functions

**QtFilePropertyDialogTabTextView** (QWidget \*parent = 0)

void **setContent** (QString content)  
 Sets the content.

**class QtFilesystemPanel** : public QWidget  
*#include <qtfilesystempanel.h>* A splitted horizontal panel of file views.

## Public Functions

**QtFilesystemPanel** (QWidget \*parent = 0)

**~QtFilesystemPanel** ()

void **addView** (bool *reinitialize* = false)

void **removeView** (int i)

*sh::ui::FileView* \***currentView** ()

int **currentViewIndex** ()

int **viewsCount** ()

```
sh::ui::FileView *view (int i)
void selectFolder (std::shared_ptr<sh::filesystem::FilesystemNode> folder, int index = -1)
QList<std::shared_ptr<sh::filesystem::FilesystemNode>> selectedNodes ()
void _emit_viewmodeChanged (int i)
void buildView (std::shared_ptr<sh::ui::qt::QtFileView> *list, sh::ui::qt::QtFileViewControl
               *viewctrl, bool isNew)
void setCustomWidget (int i, std::shared_ptr<QWidget> w)
std::shared_ptr<sh::ui::qt::QtFileView> viewwidget (int i)
```

## Signals

```
void panelFolderActivated (std::shared_ptr<sh::filesystem::FilesystemNode> folder, bool re-
                           alSwitch)
void panelCurrentViewChanged ()
void panelViewmodeChanged (int i)
void panelViewOptionChanged (int i)
void panelViewCountChanged ()
void panelSelectionChanged ()
```

## Private Members

```
Ui::QtFilesystemPanel *ui
std::shared_ptr<sh::ui::qt::QtFileView> activeView
QString activeColor
QString inactiveColor
QList<std::shared_ptr<sh::ui::qt::QtFileView>> _views
QList<QWidget*> _mainviews
QHash<QWidget*, std::shared_ptr<QWidget>> _customwidgets
QList<sh::ui::qt::QtFileViewControl*> _viewcontrols
bool _skip_eventfilter = false
```

## Friends

```
friend class ui::qt::QtMainWindow
friend class ui::FileView

class QtFileView : public std::enable_shared_from_this<QtFileView>
    #include <qtfileview.h> Abstract base class for views for the contents of one directory.
    Subclassed by sh::ui::qt::QtFileIconview, sh::ui::qt::QtFileList
```

## Public Functions

```

QtFileView ()
    Constructed only by the infrastructure and made available otherwise.

~QtFileView ()

void setBackgroundColor (QString c)

QString backgroundColor ()

QList<std::shared_ptr<sh::filesystem::FilesystemNode>> selectedNodes ()

void setSelection (const QList<std::shared_ptr<sh::filesystem::FilesystemNode>> nodes)

std::shared_ptr<sh::filesystem::FilesystemNode> node ()

QList<std::shared_ptr<sh::filesystem::FilesystemNode>> getAllVisibleNodes ()

void setSizeFormatting (sh::filesystem::SizeFormatting v)

void gotoDir (std::shared_ptr<sh::filesystem::FilesystemNode> n)

void setHiddenFilesVisible (bool value)

QThread *thread ()

void configure (sh::ui::qt::QtFileViewControl *viewctrl)

void focus ()

void setSort (int column, Qt::SortOrder order)

QObject *as_qobject ()

QWidget *as_qwidget ()

QAbstractItemView *as_qabstractitemview ()

sh::ui::qt::QtFileViewControl *control ()

```

## Private Members

```

sh::ui::qt::QtFileViewControl *_viewctrl = nullptr

QString _tmp_bgColor

QPoint _dragStartPosition

int _async_gotodir_index = 0

bool _dragIsValid = false

int _sort_column = 0

Qt::SortOrder _sort_order = Qt::AscendingOrder

```

## Friends

```
friend class sh::actions::common::ActionHistoryNavigateForward
friend class sh::actions::common::ActionHistoryNavigateBackward
friend class sh::actions::common::ActionNavigateInHistory
friend class sh::ui::qt::QtFileViewControl
class EventFilter : public QObject
```

## Public Functions

```
EventFilter (QObject *parent, QtFileView *fileview, sh::ui::qt::QtFileViewControl *file-
viewctrl)
bool eventFilter (QObject *object, QEvent *event)
```

## Private Members

```
QtFileView *fileview
sh::ui::qt::QtFileViewControl *fileviewctrl
class QtFileViewControl : public sh::ui::FileView
#include <qtfileviewcontrol.h> Qt based file view.
```

## Public Functions

```
QtFileViewControl (sh::ui::qt::QtFilesystemPanel *panel, int i)
~QtFileViewControl ()
sh::ui::ColumnDimensions *columnDimensions () override
    Returns the column dimensions handler. .
sh::tools::HistoryTracker<std::shared_ptr<const sh::filesystem::Eurl>> *historyTracker ()
                                                                    override
    Returns the history tracker. .
FileViewMode viewmode () override
    Returns the view mode (icons, list, ... ?). .
void setViewmode (FileViewMode m) override
    Sets the view mode. .
int index () override
    Returns the position of this file view within the panel. .
void gotoDir (std::shared_ptr<sh::filesystem::FilesystemNode> n) override
    Jumps to a new current directory.
    Implementations have to call the base class implementation inside.
sh::filesystem::SizeFormatting getSizeFormattingMode () override
    Returns the mode how file sizes are formatted for displaying. .
void setSizeFormattingMode (sh::filesystem::SizeFormatting mode) override
    Sets the mode how file sizes are formatted for displaying. .
```



```

bool hiddenFilesVisible () override
    Returns if hidden files are visible. .

void setHiddenFilesVisible (bool v) override
    Sets the visibility of hidden files. .

int iconDimension () override
    Returns the icon size (in pixels). .

void setIconDimension (int v) override
    Sets the icon size (in pixels). .

void setSort (int column, Qt::SortOrder order) override
    Sets how to sort this view. .

int sortColumn () override
    Returns the index of the current sort column. .

Qt::SortOrder sortOrder () override
    Returns the current sort order. .

QList<std::shared_ptr<sh::filesystem::FilesystemNode>> selectedNodes () override
    Returns the nodes which the user has selected in this fileview. .

void setSelection (const QList<std::shared_ptr<sh::filesystem::FilesystemNode>> nodes)
    override
    Sets the node selection of this fileview. .

QList<std::shared_ptr<sh::filesystem::FilesystemNode>> getAllVisibleNodes () override
    Returns a list of all nodes currently listed in this file view. .

void emit_nodesActivated (QList<std::shared_ptr<sh::filesystem::FilesystemNode>> nodes)

void emit_selectionChanged ()

std::shared_ptr<sh::filesystem::FilesystemNode> node ()
    Returns the current directory.

void reload (bool skipModel = false)
    Reloads the data inside this file view.

sh::filesystem::FilesystemModelFileviewProxy *filemodel ()
    Returns the filesystem model for this view. .

void setFilemodel (sh::filesystem::FilesystemModelFileviewProxy *model)
    Sets the filesystem model for this view. .

```

## Signals

```

void nodesActivated (QList<std::shared_ptr<sh::filesystem::FilesystemNode>> nodes)

void selectionChanged ()
    Triggered when the selection have changed.

void viewOptionChanged ()
    Triggered when some view options have changed.

```

## Private Functions

```
void setIndex (int i)
```

## Private Members

```
sh::ui::qt::QtFilesystemPanel *_panel
```

```
int i
```

```
sh::ui::ColumnDimensions *_columndimensions
```

```
sh::tools::HistoryTracker<std::shared_ptr<const sh::filesystem::Eurl>> *_historyTracker
```

```
FileViewMode _viewmode = FileViewMode::ListView
```

```
bool _hiddenfilesvisible = false
```

```
int _icondimension = 0
```

```
int _sortColumn
```

```
Qt::SortOrder _sortOrder
```

```
sh::filesystem::SizeFormatting _sizeformattingmode = sh::filesystem::SizeFormatting::SizeFormattingModePrefixes
```

## Friends

```
friend class sh::ui::qt::QtFilesystemPanel
```

```
class QtJumpBar : public QWidget
```

```
    #include <qtjumpbar.h> Button bar for navigating to places with parent-buttons and an text entry.
```

## Public Functions

```
QtJumpBar (QWidget *parent = 0)
```

```
~QtJumpBar ()
```

```
void setButtonMode ()
```

```
void setTextMode ()
```

```
bool isTextMode ()
```

```
std::shared_ptr<sh::filesystem::FilesystemNode> node ()
```

```
void setNode (std::shared_ptr<sh::filesystem::FilesystemNode> node)
```

```
QSize sizeHint () const
```

## Signals

```
void nodeChanged ()
void eurlRequested (std::shared_ptr<const sh::filesystem::Eurl> eurl)
```

## Private Functions

```
void _buildButtons ()
```

## Private Members

```
std::shared_ptr<sh::filesystem::FilesystemNode> _node = 0
Ui::QtJumpBar *ui
bool _isTextMode
```

## Private Slots

```
void on_btnOk_clicked ()
void on_btnAbort_clicked ()
```

```
class QtLinkButton : public QToolButton
    #include <qtlinkbutton.h> QToolButton with different look.
    Subclassed by sh::ui::qt::QtSearchPanelButton
```

## Public Functions

```
QtLinkButton (QWidget *parent = 0)
void setSizeByFactor (int f)
void setMenu (QStringList items, QString menuchangetxt = tr("(change)"))
void setSelectedMenuItem (int idx)
```

## Signals

```
void menuItemSelected (int idx)
```

## Private Members

```
QString _linkcolor
QStringList _menuitems
QString _menuchangetxt
```

```
class QtLogViewDialog : public sh::ui::qt::QtDialog, public sh::ui::LogViewDialog
    #include <qtlogviewdialog.h> Qt based log view dialog.
```

## Public Functions

**QtLogViewDialog** (QString *headtext*, QString *subheadtxt*, *base::LogSeverity* *minseverity*)

**~QtLogViewDialog** ()

QString **headtext** ()

Returns the header text.

QString **subheadtxt** ()

Returns the 2nd header text.

*sh::base::LogSeverity* **minimumSeverity** ()

Returns the minimum severity.

qint64 **dialogId** ()

Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

bool **isInited** ()

Returns if this dialog is initialized.

Must be called in main thread.

bool **wasAccepted** ()

Returns if this dialog was ‘accepted’ by the user (typically by clicking some ‘OK’ button).

Must be called in main thread.

void **waitClosed** ()

Wait until the user closed the dialog in some way.

May be called in any thread.

void **close** ()

Closes the dialog.

Must be called in main thread.

bool **wasClosed** ()

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \***manager** ()

Returns the *DialogManager* which hosts this dialog.

## Private Functions

void **\_reloadLog** ()

void **setMinimumSeverity** (*sh::base::LogSeverity* *minseverity*)

## Private Members

*Ui::QtLogViewDialog \*ui*

## Private Slots

void **on\_btnClose\_clicked()**

void **on\_btnSaveToFile\_clicked()**

void **on\_btnSeverity\_clicked()**

**class QtMainWindow: public QMainWindow, public *sh::ui::MainWindow***  
*#include <qtmainwindow.h>* The qt main window implementation.

## Public Functions

**QtMainWindow** (QWidget \*parent = 0)

**~QtMainWindow()**

void **initialize()** **override**

Initializes this main window. .

std::shared\_ptr<*DialogManager*> **dialogManager()** **override**

Returns the *DialogManager* which creates and drives *Dialogs* for this main window.

You typically should not need to use it directly.

int **fileViewCount()** **override**

Returns the current number of file views. .

void **addFileView** (bool *reinitialize* = false) **override**

Adds a new file view. .

void **removeFileView** (int *i*) **override**

Removes a file view. .

*sh::ui::FileView* \***currentFileView()** **override**

Returns the file view which is currently active (i.e. focussed). .

*sh::ui::FileView* \***getFileView** (int *i*) **override**

Returns a file view by position index. .

void **jumpToNode** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> *node*) **override**

Let the current view jump to another location. .

void **setTitlePattern** (QString *value*) **override**

Sets the window title pattern. .

void **setTreeVisible** (bool *v*) **override**

Sets the visibility of the directory tree. .

void **setFileDetailsPanelVisible** (bool *v*) **override**

Sets the visibility of the file details panel. .

std::shared\_ptr<*sh::filesystem::FilesystemNode*> **currentDirectory()** **override**

Gets the *sh::filesystem::FilesystemNode* selected in the current view. .

```
std::shared_ptr<sh::ui::ActionExecutionInfoPanel> addInfoPanel (int position,
                                                             sh::actions::ActionExecutionInfo
                                                             *info, QColor color =
                                                             QColor()) override

    Creates a new action execution panel.

    Let this smart pointer die for removing it.

void jumpbarSetTextMode () override
    Sets the jumpbar to text input mode. .

void jumpbarSetButtonMode () override
    Sets the jumpbar to button mode. .

bool jumpbarIsTextMode () override
    Returns if the jumpbar is in text input mode. .

QString toolbarPosition () override
    Returns the current toolbar position. .

void setToolbarPosition (QString pos) override
    Sets the toolbar position. .

QString detailPanelPosition () override
    Returns the current detail panel position. .

void setDetailPanelPosition (QString pos) override
    Sets the detail panel position. .

std::shared_ptr<AboutDialog> showAboutDialog () override
    Shows and returns an 'About' dialog. .

std::shared_ptr<ManageProfilesDialog> showManageProfilesDialog () override
    Shows and returns a 'Manage Saved Settings' dialog. .

std::shared_ptr<OpenWithDialog> showOpenWithDialog (std::shared_ptr<sh::filesystem::FilesystemNode>
                                                    node,
                                                    QList<std::shared_ptr<sh::tools::filetypes::OpenMethod>>
                                                    openMethods) override

    Shows and returns a 'Open With' dialog. .

std::shared_ptr<StoreProfileDialog> showStoreProfileDialog (std::shared_ptr<const
                                                            sh::filesystem::Eurl> eurl)
                                                            override

    Shows and returns a 'Save Settings' dialog. .

std::shared_ptr<TuningDialog> showTuningDialog () override
    Shows and returns a 'Tuning' dialog. .

std::shared_ptr<LogViewDialog> showLogViewDialog (QString headtext = QString(),
                                                  QString subheadtxt = QString(),
                                                  sh::base::LogSeverity minseverity =
                                                  sh::base::LogSeverity::_DEFAULT)
                                                  override

    Shows and returns a 'Log' dialog. .

std::shared_ptr<ManageBookmarksDialog> showManageBookmarksDialog () override
    Shows and returns a 'Manage Bookmarks' dialog. .

std::shared_ptr<FilePropertyDialog> showFilePropertyDialog (QList<std::shared_ptr<sh::filesystem::FilesystemNode>
                                                            nodes) override

    Shows and returns a 'File Properties' dialog. .
```

```

std::shared_ptr<ActionExecutionInfoDialog> createActionExecutionInfoDialog (sh::actions::ActionExecutionInfo
                                                                    *info)
                                                                    override

    Creates an action execution dialog. .

void handleCloseAppRejected (QString rejectmsg) override
    React on a rejected application close request (with some feedback message). .

void handleClosed () override
    Do some cleanup steps just when closing starts. Implementations must call base implementation!

sh::ui::qt::QtUIStyle *uiStyle ()
    Gets a QtUIStyle instance, which helps for common ui styling.

void _initialize (sh::base::SingletonInitializer *singletonInitializer)
    Initializes the main window. For custom initialization logic, see MainWindow.initialize. .

void fileViewsReloadAll ()
    Reload all content in each file view.

void onFileViewOptionsChanged (std::function<void> int
    > fct, QObject *owner = 0) Sets a handler for some view options in a file view changed (optionally
    bound to an owner lifetime).

void onCurrentFileViewChanged (std::function<void>
    > fct, QObject *owner = 0) Sets a handler for the currently active file view changed (optionally bound to
    an owner lifetime).

void onFileViewCountChanged (std::function<void>
    > fct, QObject *owner = 0) Sets a handler for the number of file views changed (optionally bound to an
    owner lifetime).

void onCurrentDirectoryChanged (std::function<void>
    > fct, QObject *owner = 0) Sets a handler for the current directory in the active file view changed (op-
    tionally bound to an owner lifetime).

void onGlobalViewOptionsChanged (std::function<void>
    > fct, QObject *owner = 0) Sets a handler for some global view options changed (optionally bound to an
    owner lifetime).

void onCurrentProfileChanged (std::function<void>
    > fct, QObject *owner = 0) Sets a handler for the current profile changed (optionally bound to an owner
    lifetime).

void jumpToEurl (std::shared_ptr<const sh::filesystem::Eurl> eurl)
    Let the current view jump to another location. .

QString titlePattern ()
    Returns the window title pattern.

void setCurrentProfile (QString profile)
    Sets the current profile. .

QString currentProfile ()
    Returns the current profile.

void reloadProfile ()
    Reloads the profile data.

bool treeVisible ()
    Returns the visibility of the directory tree.

```

void **setTreeSticky** (bool v)  
Sets if the directory tree should follow the active file view. .

bool **treeSticky** ()  
Returns if the directory tree follows the active file view.

bool **fileDetailsPanelVisible** ()  
Returns the visibility of the file details panel.

std::shared\_ptr<*sh::ui::ActionExecutionInfoPanel*> **addInfoPanel** (*sh::actions::ActionExecutionInfo* \*info, QColor color = QColor())  
Creates a new action execution panel.  
Let this smart pointer die for removing it.

std::shared\_ptr<*ActionExecutionInfoPanel*> **showErrorPanel** ()  
Shows an error panel.

void **\_closeDirectly** ()  
Directly begin application shutdown without checks.

bool **closeApp** ()  
Check if the application may shut down now, and if so, initiate it.

bool **isCloseable** (QString \*msg = nullptr)  
Check if the application may shut down now.

bool **isClosing** ()  
Returns if the application is currently closing.

## Public Static Functions

bool **tryCreateMainWindow** (*MainWindow* \*&mainWindow)  
*QtMainWindow* \*mainWindow ()

void **setMainWindow** (*MainWindow* \*mainWindow)  
Sets the main window instance. .

bool **isReady** ()  
Returns if this process ui is ready (i.e. is created or runs headless).

bool **runsHeadless** ()  
Returns if this process runs headless, i.e. without any ui, just for a background process.

void **\_closeDirectly** (*sh::base::SingletonInitializer* \*singletonInitializer)

QString **uiMode** ()  
Returns the UI mode (e.g. qt, web).



## Private Functions

```

void _setupGlobalShortcut (std::shared_ptr<sh::actions::AbstractActionItem> action)
void _setupGlobalShortcut_recursive (std::shared_ptr<sh::actions::SubmenuItem>
                                     submenu)
void _addPermanentActionToToolbar (std::shared_ptr<sh::actions::SubmenuItem> a,
                                   bool rightSide = false, bool preventDefaultAction =
                                   false)
void _refreshToolbar (std::shared_ptr<sh::actions::ActionInstantiation>)
void _jumpToNode (std::shared_ptr<sh::filesystem::FilesystemNode> node, int skip)
void _jumpToIndex (QModelIndex idx, int skip)
void _newToStatusbar_helper (sh::ui::qt::QtActionExecutionInfoPanel *w)
void _refresh_detailthumbnailvisibility ()
void _thumbnail_resized ()
void _resizethumbnail ()
void _detailpanel_resized ()
void _resizedetailpanel ()
void requestDetailThumbnail (bool force = true)
void setStatusbarVisibility (bool v)

```

## Private Members

```

Ui::QtMainWindow *ui
sh::ui::qt::QtFilesystemPanel *fspanel
sh::filesystem::FilesystemModelDirectoryTreeProxy *treemodel = 0
sh::ui::qt::QtJumpBar *jumpbar
QTimer _statusbar_timer
int _statusbar_fullheight
int _statusbar_targetheight = 0
QSet<sh::ui::qt::QtActionExecutionInfoPanel*> _statusbar_childs
std::shared_ptr<sh::filesystem::FilesystemNode> _currentDirectory = 0
bool _request_detailthumbnail_skip = false
int _thumbnailwidth = 0
Qt::Orientation _myorientation = Qt::Horizontal
sh::ui::qt::QtUIStyle *_uistyle = 0
QAction *_toolbarLeftRightSplitSeparator
QHash<QString, std::shared_ptr<sh::actions::common::ActionGroups>> actionGroupsActions
QList<QtToolBarButtonHandler*> toolbarButtonHandlers
std::shared_ptr<QtDialogManager> _dialogManager

```

### Private Slots

```
void slot_detailbar_moved ()
void slot_toolbar_moved ()
void slot_statusbartimer ()
void slot_treeview_collapsedexpanded (const QModelIndex &index)
```

### Private Static Attributes

```
QtMainWindow *_qtMainWindow = nullptr
```

### Friends

```
friend class QtToolbarButton
friend class QtFilesystemPanel
class MyFlexibleLabel : public QLabel
    #include <qtmainwindow.h> Needed internally in main window in order to have a label which really
    does not request any sizes.
```

### Public Functions

```
MyFlexibleLabel (QWidget *parent = 0)
QSize sizeHint () const override
class QtManageBookmarksDialog : public sh::ui::qt::QtDialog, public sh::ui::ManageBookmarksDialog
    #include <qtmanagebookmarksdialog.h> Qt based manage bookmarks dialog.
```

### Public Functions

```
QtManageBookmarksDialog ()
~QtManageBookmarksDialog ()
qint64 dialogId ()
    Returns the dialog id.
    Each instance has an id unique in the complete Shallot process lifetime.
    Must be called in main thread.
bool isInitiated ()
    Returns if this dialog is initialized.
    Must be called in main thread.
bool wasAccepted ()
    Returns if this dialog was ‘accepted’ by the user (typically by clicking some ‘OK’ button).
    Must be called in main thread.
void waitClosed ()
    Wait until the user closed the dialog in some way.
    May be called in any thread.
```

```
void close ()
    Closes the dialog.

    Must be called in main thread.

bool wasClosed ()
    Returns if this dialog was closed.

    Must be called in main thread.

DialogManager *manager ()
    Returns the DialogManager which hosts this dialog.
```

### Private Functions

```
void readBookmarks ()
void _refresh_values ()
void _selectbookmark (QString id)
void _selectfolder (QStringList name)
```

### Private Members

```
Ui::QtManageBookmarksDialog *ui
```

### Private Slots

```
void on_btnClose_clicked ()
void on_treeWidget_itemSelectionChanged ()
void on_btnDelete_clicked ()
void on_edtLabel_textChanged (const QString &arg1)
void on_edtLocation_textChanged (const QString &arg1)
void on_btnStore_clicked ()
void on_btnNew_clicked ()
void on_btnUp_clicked ()
void on_btnDown_clicked ()
void on_btnMove_clicked ()

class QtManageProfilesDialog: public sh::ui::qt::QtDialog, public sh::ui::ManageProfilesDialog
    #include <qtmanageprofilesdialog.h> Qt based manage saved settings dialog.
```

## Public Functions

**QtManageProfilesDialog()**

**~QtManageProfilesDialog()**

**qint64 dialogId()**

Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

**bool isInitiated()**

Returns if this dialog is initialized.

Must be called in main thread.

**bool wasAccepted()**

Returns if this dialog was 'accepted' by the user (typically by clicking some 'OK' button).

Must be called in main thread.

**void waitClosed()**

Wait until the user closed the dialog in some way.

May be called in any thread.

**void close()**

Closes the dialog.

Must be called in main thread.

**bool wasClosed()**

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \***manager()**

Returns the *DialogManager* which hosts this dialog.

## Private Functions

**void \_createProfileList()**

**void \_createSettingsList()**

**void \_createNodesList()**

## Private Members

*Ui::QtManageProfilesDialog* \***ui**

*QStringListModel* \***\_model**

*sh::settings::ProfileNode* \***\_currentNode = 0**

*QHash<QListWidgetItem\*, QString>* **\_nodes**

## Private Slots

```
void on_comboBox_currentTextChanged (const QString &arg1)
void on_btnClose_clicked ()
void on_btnDelNode_clicked ()
void on_btnDelProfile_clicked ()
void slot_listViewactivated (QListWidgetItem*, QListWidgetItem*)

class QtOpenWithDialog : public sh::ui::qt::QtDialog, public sh::ui::OpenWithDialog
#include <qopenwithdialog.h> Qt based open with dialog.
```

## Public Functions

```
QtOpenWithDialog (std::shared_ptr<sh::filesystem::FilesystemNode> node,
                  QList<std::shared_ptr<sh::tools::filetypes::OpenMethod>> openMethods)
~QtOpenWithDialog ()
void setProgramList (QList<std::shared_ptr<sh::tools::filetypes::OpenMethod>> openMethods)
std::shared_ptr<sh::tools::filetypes::OpenMethod> chosenMethod () override
    Returns the chosen open-method (program for opening the file).
bool rememberForMimeType () override
    Returns if the chosen method is checked to be remembered for the same mime-type in the future.
std::shared_ptr<const sh::filesystem::Eurl> rememberForDirectory () override
    If it's checked for the chosen method to be remembered for some subdirectory in the future, it returns
    the Eurl of this subdirectory, otherwise nullptr.
bool rememberForFile () override
    Returns if the chosen method is checked to be remembered for the same file in the future.
std::shared_ptr<sh::filesystem::FilesystemNode> node ()
    Returns the file node which is later to be opened.
QList<std::shared_ptr<sh::tools::filetypes::OpenMethod>> openMethods ()
    Returns the open-methods (programs for opening the file) to present for choice.
qint64 dialogId ()
    Returns the dialog id.
    Each instance has an id unique in the complete Shallot process lifetime.
    Must be called in main thread.
bool isInitiated ()
    Returns if this dialog is initialized.
    Must be called in main thread.
bool wasAccepted ()
    Returns if this dialog was 'accepted' by the user (typically by clicking some 'OK' button).
    Must be called in main thread.
```

void **waitClosed** ()  
Wait until the user closed the dialog in some way.  
May be called in any thread.

void **close** ()  
Closes the dialog.  
Must be called in main thread.

bool **wasClosed** ()  
Returns if this dialog was closed.  
Must be called in main thread.

*DialogManager* \***manager** ()  
Returns the *DialogManager* which hosts this dialog.

### Private Members

std::shared\_ptr<*sh::tools::filetypes::OpenMethod*> **\_chosenMethod**  
std::shared\_ptr<const *sh::filesystem::Eurl*> **\_rememberfordirectory**  
QAbstractItemModel \* **\_listmodel**  
*Ui::QtOpenWithDialog* \***ui**

### Private Slots

void **on\_btnCancel\_clicked** ()  
void **on\_btnOk\_clicked** ()  
void **on\_btnSelectOther\_clicked** ()  
void **on\_listView\_activated** (const QModelIndex &*index*)  
void **on\_btnAnotherAncestor\_clicked** ()  
void **on\_chkRememberDir\_toggled** (bool *checked*)

**class QtOpenWithDialogModel : public QStringListModel**  
*#include <qopenwithdialog.h>* Used internally in OpenWithDialogModel.

### Public Functions

**QtOpenWithDialogModel** (QList<std::shared\_ptr<*sh::tools::filetypes::OpenMethod*>> *meth-*  
*ods*, QObject \**parent* = 0)  
QVariant **data** (const QModelIndex &*index*, int *role*) const

## Private Members

`QList<std::shared_ptr<sh::tools::filetypes::OpenMethod>> _methods`

**class** `QtSearchPanel` : **public** `QWidget`  
*#include <qtsearchpanel.h>* Search panel for usage in the Qt main window.

## Public Functions

`QtSearchPanel` (`std::shared_ptr<sh::filesystem::FilesystemNode> node`, `QWidget *parent = 0`)

`~QtSearchPanel` ()

## Friends

**friend class** `QtSearchPanelConfiguration`

`template<class T1, class T2>`

**class** `QtSearchPanelAbstractEditor` : **public** `T1`, **public** `T2`  
*#include <qtsearchpanelconfiguration.h>* Helper for other Qt based search panel editors.

## Public Functions

`QtSearchPanelAbstractEditor` (`QWidget *parent = 0`)

`bool isEnabled` ()

`void setEnabled` (`bool v`)

**class** `QtSearchPanelButton` : **public** `sh::ui::qt::QtLinkButton`, **public** `sh::ui::SearchPanelButton`  
*#include <qtsearchpanelconfiguration.h>* Qt based [SearchPanelButton](#).

## Public Functions

`QtSearchPanelButton` (`QWidget *parent = 0`)

`void setButtonText` (`QString txt`)  
 Sets the button text. .

`void setMenuSelection` (`int i`)  
 Sets the currently selected menu item (for buttons with menus). .

`void setSizeByFactor` (`int f`)

`void setMenu` (`QStringList items`, `QString menuchangetxt = tr("(change)")`)

`void setSelectedMenuItem` (`int idx`)

## Signals

void **menuItemSelected** (int *idx*)

```
class QtSearchPanelConfiguration : public sh::ui::SearchPanelConfiguration
#include <qtsearchpanelconfiguration.h> Search panel configuration for usage in a Qt ui.
```

## Public Functions

**QtSearchPanelConfiguration** (*QtSearchPanel* \**owner*, QHBoxLayout \**editors*)

*SearchPanelButton* \***addMenuButton** (QString *text*, QStringList *menu*, std::function<void> int  
> *onchanged*) Adds and returns a button for a menu. .

*SearchPanelButton* \***addActionButton** (QString *text*, std::function<void>  
> *action*) Adds and returns a button for an action. .

*SearchPanelTextEditor* \***addTextEditor** ()  
Adds and returns a text editor. .

*SearchPanelDateTimeEditor* \***addDateTimeEditor** ()  
Adds and returns a date/time editor. .

*SearchPanelLabelEditor* \***addLabel** ()  
Adds and returns a label. .

*SearchPanelSpacerEditor* \***addSpacer** ()  
Adds and returns a spacer. .

*SearchPanelAbstractEditor* \***getEditorAt** (int *i*)  
Returns the editor widget at a given position. .

void **onDestroyed** (std::function<void>  
> *fcn*) *QObject* \**owner* = 0 Sets a handler for panel removal (optionally bound to an owner lifetime).

## Public Members

*\_SearchPanelConfiguration\_HelperQObject* **myqobject**

## Private Members

*QtSearchPanel* \***owner**

QHBoxLayout \***editors**

```
class QtSearchPanelDateTimeEditor : public sh::ui::qt::QtSearchPanelAbstractEditor<QDateTimeEdit, SearchPa
#include <qtsearchpanelconfiguration.h> Qt based SearchPanelDateTimeEditor.
```



## Public Functions

```

QtSearchPanelDateTimeEditor (QWidget *parent = 0)

QDateTime datetime ()
    Returns the selected date/time. .

void setDatetime (QDateTime s)
    Sets the selected date/time.

bool isWidgetEnabled ()

void setWidgetEnabled (bool v)

class QtSearchPanelLabelEditor : public sh::ui::qt::QtSearchPanelAbstractEditor<QLabel, sh::ui::SearchPanelLa
    #include <qtsearchpanelconfiguration.h> Qt based SearchPanelLabelEditor.

```

## Public Functions

```

QtSearchPanelLabelEditor (QWidget *parent = 0)

QString textContent ()
    Returns the text content. .

void setTextContent (QString s)
    Sets the text content. .

SearchPanelAbstractEditor *focusProxyEditor ()
    Returns the focus proxy widget (which focus gets redirected to in some situations). .

void setFocusProxyEditor (SearchPanelAbstractEditor *w)
    Sets the focus proxy widget (which focus gets redirected to in some situations). .

bool isWidgetEnabled ()

void setWidgetEnabled (bool v)

class QtSearchPanelSpacerEditor : public sh::ui::qt::QtSearchPanelAbstractEditor<QLabel, sh::ui::SearchPanelS
    #include <qtsearchpanelconfiguration.h> Qt based SearchPanelSpacerEditor.

```

## Public Functions

```

QtSearchPanelSpacerEditor (QWidget *parent = 0)

bool isWidgetEnabled ()

void setWidgetEnabled (bool v)

class QtSearchPanelTextEditor : public sh::ui::qt::QtSearchPanelAbstractEditor<QLineEdit, sh::ui::SearchPanelT
    #include <qtsearchpanelconfiguration.h> Qt based SearchPanelTextEditor.

```

### Public Functions

```
QtSearchPanelTextEditor (QWidget *parent = 0)

QString textContent ()
    Returns the text content. .

void setTextContent (QString s)
    Sets the text content. .

QString placeholderDescription ()
    Returns the placeholder description text (visible if field is empty). .

void setPlaceholderDescription (QString s)
    Sets the placeholder description text (visible if field is empty). .

bool isWidgetEnabled ()

void setWidgetEnabled (bool v)

class QtSettingUIFrame : public QFrame
    #include <qtsettinguiframe.h> User interface for one handling one setting.
```

### Public Functions

```
QtSettingUIFrame (sh::settings::Setting *setting, bool withcheckbox, QString displayvalue,
                  QString additionalCheck, QVariant additionalCheckValue, QWidget *parent = 0)

bool isChecked ()

void setChecked (bool val)

void setAdditionalCheckVisible (bool v)

QVariant additionalCheckValue ()
```

### Private Members

```
QCheckBox *_checkbox = 0

QCheckBox *_additionalcheckbox = 0

QLabel *_additionalcheckboxlabel = 0

QWidget *_additionalcheckwidget = 0

QVariant _additionalCheckValue

class QtStoreProfileDialog : public sh::ui::qt::QtDialog, public sh::ui::StoreProfileDialog
    #include <qtstoreprofiledialog.h> Qt based save settings dialog.
```

## Public Functions

**QtStoreProfileDialog** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*)

**~QtStoreProfileDialog** ()

**QList<*sh::ui::StoreProfileDialog::SettingEntry*> checkedSettings** () **override**

Returns the list of settings the user has checked to store.

**QString profileName** () **override**

Returns the profile the user has chosen to store settings for.

**bool withSubDirectories** () **override**

Returns if the user has chosen to apply the checked settings also for subdirectories.

**QStringList inheritsFrom** () **override**

Returns the list of profiles the user has chosen to inherit from.

**bool hasGlobal** () **override**

Returns if the user has chosen to apply the checked settings for each directory (instead of the current directory).

**std::shared\_ptr<const *sh::filesystem::Eurl*> eurl** ()

Returns the current directory this dialog shall work on.

**qint64 dialogId** ()

Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

**bool isInitd** ()

Returns if this dialog is initialized.

Must be called in main thread.

**bool wasAccepted** ()

Returns if this dialog was 'accepted' by the user (typically by clicking some 'OK' button).

Must be called in main thread.

**void waitClosed** ()

Wait until the user closed the dialog in some way.

May be called in any thread.

**void close** ()

Closes the dialog.

Must be called in main thread.

**bool wasClosed** ()

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \***manager** ()

Returns the *DialogManager* which hosts this dialog.

### Private Functions

```
void __settingsLevel (int level, bool persist = true)
void __setglobal (bool v)
```

### Private Members

```
QHash<sh::ui::qt::QtSettingUIFrame*, std::shared_ptr<sh::settings::Setting>> __widgets
QHash<int, QLabel*> __headerlabels
QStringList __inheritProfileNameList
bool __global = false
Ui::QtStoreProfileDialog *ui
int __level
```

### Private Slots

```
void on_btnAddProf_clicked ()
void on_btnGlobal_clicked ()
void on_btn22_clicked ()
void on_toolButton_2_clicked ()
void on_toolButton_clicked ()

class QtToolBarButton : public QToolButton
    #include <qttoolbarbutton.h> A qt toolbar button implementation with optional submenu.
```

### Public Types

```
enum Position
    Values:
        enumerator LEFT
        enumerator RIGHT
        enumerator TOP
        enumerator BOTTOM
```

### Public Functions

```
QtToolBarButton (QWidget *parent = 0)
void setText (const QString &text)
QString text () const
void setIcon (const QIcon &icon)
QIcon icon ()
QSize sizeHint () const
```

```

void setButtonText (QString v)
QString buttonText ()
void setButtonIcon (QIcon v)
QIcon buttonIcon ()
void setButtonEnabled (bool v)
bool isButtonEnabled ()
bool hasExpander ()
void setSubmenu (QtActionMenu *menu)
QtActionMenu *submenu ()
void setLocation (Position location)
void setClickAction (std::function<void>
    > action)
void unsetClickAction ()
void trigger ()
void openMenu ()

```

### Private Functions

```

void _calcDims ()
void computeArrowAndDividerLineCoordinates ()

```

### Private Members

```

QString _text
QString _text1
QString _textWithoutMnemonic1
QString _text2
QString _textWithoutMnemonic2
QIcon _icon
bool _drawIconOnly = false
bool _hovered = false
bool _arrowhovered = false
sh::ui::qt::QtActionMenu * _menu = 0
std::function<void ()> _clickAction
Position location
int tw1
int tworig1
int tw2

```

```
int tworig2
int twmax
int tworigmax
int th
int thorig
int iw
int ih
int iedim
int ew
int eh
const int textpad = 20
const int opad = 3
int expx1
int expx2
int expy1
int expy2
int xt
int yt1
int yt2
int xi
int yi
QPainterPath arrow
bool _drawmnemonics = false
```

### Private Slots

```
void slot_clicked()
```

### Friends

```
friend class sh::actions::ActionsManager
```

```
class QtToolBarButtonHandler
```

*#include <qttoolbarbuttonhandler.h>* A handler which reflects the *sh::actions* objects to a graphical toolbar button (and keeps that up-to-date).

## Public Functions

```
QtToolbarButtonHandler (std::shared_ptr<sh::actions::SubmenuItem> action, bool
                        preventDefaultAction, QtToolbarButton *button, QAction *qac-
                        tion)
~QtToolbarButtonHandler ()
```

## Private Functions

```
void __updateSubmenu ()
```

## Private Members

```
std::shared_ptr<sh::actions::SubmenuItem> action
QtActionMenu menu
bool preventDefaultAction
QtToolbarButton *button
```

## Private Static Functions

```
void __applyPropertiesToButton (sh::actions::SubmenuItem *action, QtToolbarButton
                                *button, QAction *qaction)
```

```
class QtTuningDialog: public sh::ui::qt::QtDialog, public sh::ui::TuningDialog
    #include <qttuningdialog.h> Qt based tuning dialog.
```

## Public Functions

```
QtTuningDialog ()
```

```
~QtTuningDialog ()
```

```
qint64 dialogId ()
    Returns the dialog id.
```

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

```
bool isInitied ()
    Returns if this dialog is initialized.
```

Must be called in main thread.

```
bool wasAccepted ()
    Returns if this dialog was ‘accepted’ by the user (typically by clicking some ‘OK’ button).
```

Must be called in main thread.

```
void waitClosed ()
    Wait until the user closed the dialog in some way.
```

May be called in any thread.

void **close** ()  
Closes the dialog.  
Must be called in main thread.

bool **wasClosed** ()  
Returns if this dialog was closed.  
Must be called in main thread.

*DialogManager* \***manager** ()  
Returns the *DialogManager* which hosts this dialog.

## Private Types

**typedef** QPair<QString, QWidget\*> **BoxWidgetByDesc**

## Private Functions

QString **\_changedlglabel** (std::shared\_ptr<*sh::configuration::ConfigurationValue*> cv)  
void **do\_change** (std::shared\_ptr<*sh::configuration::ConfigurationValue*> cv)  
void **filterVisibility** (QString s)

## Private Members

*Ui::QtTuningDialog* \***ui**  
QList<*BoxWidgetByDesc*> **boxWidgetsByDesc**

## Private Slots

void **on\_btnCancel\_clicked** ()  
void **on\_lineEditSearchInt\_textChanged** (const QString &*arg1*)  
void **on\_lineEditSearchExt\_textChanged** (const QString &*arg1*)  
void **on\_lineEditSearchBehav\_textChanged** (const QString &*arg1*)  
void **on\_lineEditSearchAppear\_textChanged** (const QString &*arg1*)  
void **on\_tabWidget\_currentChanged** (int *index*)

**class** **QtUIStyle**  
#include <*qtuistyle.h*> Methods for applying the Shallot visible style.  
See *sh::ui::qt::QtMainWindow* about how to get an instance.



## Public Functions

**QtUIStyle** (*sh::ui::qt::QtMainWindow \*mainwnd*)

Constructed only by the infrastructure and made available otherwise.

QColor **windowColor** ()

QColor **backgroundColor** ()

QColor **inactiveBackgroundColor** ()

QColor **windowColorHighlighted** ()

QColor **windowColorHalfHighlighted** ()

QColor **windowColorDarkened** ()

QColor **windowColorLikeTitlebar** ()

QColor **linkColor** ()

QColor **titlebarColor** ()

QColor **foregroundColor** ()

QColor **foregroundColorLighter1** ()

QColor **foregroundColorLighter2** ()

int **fontSizeInPt** (int *r* = 100)

QColor **mix** (float *nI*, QColor *cI*, QColor *c2*)

*Sheet* **createSheet** ()

## Private Members

*sh::ui::qt::QtMainWindow* \*\_mainWindow

**class Sheet**

#include <qtuistyle.h>

## Public Functions

QString **sheet** ()

*Sheet* **customcss** (QString *css*)

*Sheet* **fontsize\_byFactor** (int *f*)

*Sheet* **fontsize\_header** ()

*Sheet* **fontsize\_smaller** ()

*Sheet* **textcolor** (QColor *color*)

*Sheet* **textcolor\_lighter1** ()

*Sheet* **textcolor\_lighter2** ()

*Sheet* **background** (QColor *color*, bool *restrictQWidget* = true)

*Sheet* **background\_highlighted** (bool *restrictQWidget* = true)

*Sheet* **background\_halfhighlighted** (bool *restrictQWidget* = true)

```
Sheet background_darkened (bool restrictQWidget = true)
Sheet background_liketitlebar (bool restrictQWidget = true)
Sheet applyTo (QWidget *widget)
Sheet bold ()
```

### Private Functions

```
Sheet (QtUIStyle *style, QString sheet)
```

### Private Members

```
QString _sheet
QtUIStyle *_style
```

### Friends

```
friend class QtUIStyle
```

### namespace feedbackpanels

Qt user interface parts for user feedback in action execution dialogs.

Implementations of *sh::ui::qt::feedbackpanels::FeedbackPanel* are used for implementing parts of *sh::ui::ActionExecutionInfoDialog*.

### Typedefs

```
typedef QPair<QString, QWidget*> GridFormRow
```

```
class Credentials: public sh::ui::qt::feedbackpanels::FeedbackPanel
    #include <credentials.h> FeedbackPanel for user login credentials (password based).
```

### Public Functions

```
Credentials (QString domain, QString username, QString password, QString text, bool
    showDomain, bool showUsername, bool showPassword, bool showAnony-
    mous, bool showRemember)

~Credentials ()

void cancelRequested ()

bool isClosed ()

void waitUntilClosed ()

int preferredHeight ()
```

## Public Members

QString **domain**  
 QString **username**  
 QString **password**  
 bool **anonymous**  
 bool **remember**  
 bool **accepted**

## Signals

void **wasClosed** ()

## Private Members

*Ui::*Credentials \***ui**

## Private Slots

void **on\_chkAnonymous\_toggled** (bool *checked*)  
 void **on\_pushButton\_3\_clicked** ()  
 void **on\_pushButton\_4\_clicked** ()

**class FeedbackPanel : public QWidget**  
*#include <feedbackpanel.h>* Abstract base class for a Qt helper widget used in action user feedback.  
 Subclassed by *sh::ui::qt::feedbackpanels::Credentials*, *sh::ui::qt::feedbackpanels::GridForm*,  
*sh::ui::qt::feedbackpanels::MsgBox*, *sh::ui::qt::feedbackpanels::UnixPermissions*

## Public Functions

**FeedbackPanel** (QWidget \**parent* = 0)  
 bool **isClosed** ()  
 void **waitUntilClosed** ()  
 int **preferredHeight** ()  
 void **cancelRequested** () = 0

### Signals

void **wasClosed** ()

### Private Members

QMutex **\_mutex**

QWaitCondition **\_closedcondition**

bool **\_closed** = false

**class GridForm**: public *sh::ui::qt::feedbackpanels::FeedbackPanel*  
*#include <gridform.h> FeedbackPanel* for grid based forms.

### Public Functions

**GridForm** (QString *text*, QList<*sh::ui::qt::feedbackpanels::GridFormRow*> *form*, QStringList  
*answers*, int *defaultanswer* = -1, int *cancelanswer* = -1, QWidget *\*parent* = 0)

**~GridForm** ()

int **preferredHeight** ()

void **cancelRequested** ()

bool **isClosed** ()

void **waitUntilClosed** ()

### Public Members

int **answer** = -1

### Signals

void **wasClosed** ()

### Private Members

*Ui::*GridForm **\*ui**

QHash<QPushButton\*, int> **btn2index**

QPushButton **\*\_cancelBtn** = 0

QPushButton **\*\_defaultBtn** = 0

### Private Slots

```
void slot_btnclicked()
```

```
class GridFormInnerSimpleChooser : public QWidget
#include <gridformminnersimplechooser.h> Helper widget for a GridForm.
```

### Public Functions

```
GridFormInnerSimpleChooser(sh::actions::ActionExecutionUserFeedback::Choices
                           *choices, QWidget *parent = 0)
```

```
~GridFormInnerSimpleChooser()
```

### Public Members

```
int answer = -1
```

```
QString text
```

### Private Functions

```
bool eventFilter (QObject *obj, QEvent *event)
```

### Private Members

```
QLineEdit *lineedit
```

```
sh::actions::ActionExecutionUserFeedback::Choices *choices
```

```
QVBoxLayout *mylayout
```

```
sh::actions::ActionExecutionUserFeedback::Choice *currentchoice = 0
```

### Private Slots

```
void slot_chosen (int id)
```

```
void slot_textedited (QString t)
```

```
class MsgBox : public sh::ui::qt::feedbackpanels::FeedbackPanel
#include <msgbox.h> FeedbackPanel for showing a message, some answer buttons, and optionally a
text input box.
```

### Public Functions

```
MsgBox (QString question, QList<QString> answers, QString icon = "", QString withInputBox =
        QString(), bool inputBoxMultiline = false, int defaultanswer = -1, int cancelanswer = -
        1, int valuePreselectFrom = -1, int valuePreselectTo = -1, QList<QString> answericons
        = QList<QString>(), QWidget *parent = 0)
```

```
~MsgBox()
```

```
void cancelRequested()
```

```
bool isClosed()
```

```
void waitUntilClosed()
```

### Public Members

```
int answer = -1
```

```
QString text
```

### Signals

```
void wasClosed()
```

### Private Members

```
Ui::MsgBox *ui
```

```
QHash<QPushButton*, int> btn2index
```

```
QWidget *_initialFocusWidget = 0
```

```
QPushButton *_cancelBtn = 0
```

```
QPushButton *_defaultBtn = 0
```

```
bool _inputBoxMultiline
```

```
int _valuePreselectFrom
```

```
int _valuePreselectTo
```

### Private Slots

```
void slot_btnclicked()
```

```
class UnixPermissions : public sh::ui::qt::feedbackpanels::FeedbackPanel  
    #include <unixpermissions.h> FeedbackPanel for setting one set of unix filesystem permissions.
```

### Public Functions

```
UnixPermissions (QWidget *parent = 0)
```

```
~UnixPermissions ()
```

```
void cancelRequested()
```

```
void setflags (bool userMayRead, bool userMayWrite, bool userMayExecute, bool group-  
    MayRead, bool groupMayWrite, bool groupMayExecute, bool othersMayRead,  
    bool othersMayWrite, bool othersMayExecute, bool sticky, bool setuid, bool  
    setgid)
```

```
void setusers (QStringList users, QString selecteduser)
```

```
void setgroups (QStringList groups, QString selectedgroup)
```

```
bool isClosed()
```

```
void waitUntilClosed()
```

```
int preferredHeight()
```

### Public Members

bool **accepted**  
bool **userMayRead**  
bool **userMayWrite**  
bool **userMayExecute**  
bool **groupMayRead**  
bool **groupMayWrite**  
bool **groupMayExecute**  
bool **othersMayRead**  
bool **othersMayWrite**  
bool **othersMayExecute**  
bool **sticky**  
bool **setuid**  
bool **setgid**  
QString **user**  
QString **group**

### Signals

void **wasClosed**()

### Private Members

*Ui::*UnixPermissions \***ui**

### Private Slots

void **on\_pushButton\_clicked**()  
void **on\_pushButton\_2\_clicked**()

## 10.2.27 Namespace *sh::ui::qt::feedbackpanels*

**namespace** *sh::ui::qt::feedbackpanels*

Qt user interface parts for user feedback in action execution dialogs.

Implementations of *sh::ui::qt::feedbackpanels::FeedbackPanel* are used for implementing parts of *sh::ui::ActionExecutionInfoDialog*.

## Typedefs

```
typedef QPair<QString, QWidget*> GridFormRow
```

```
class Credentials : public sh::ui::qt::feedbackpanels::FeedbackPanel  
    #include <credentials.h> FeedbackPanel for user login credentials (password based).
```

## Public Functions

```
Credentials (QString domain, QString username, QString password, QString text, bool show-  
             Domain, bool showUsername, bool showPassword, bool showAnonymous, bool  
             showRemember)
```

```
~Credentials ()
```

```
void cancelRequested ()
```

```
bool isClosed ()
```

```
void waitUntilClosed ()
```

```
int preferredHeight ()
```

## Public Members

```
QString domain
```

```
QString username
```

```
QString password
```

```
bool anonymous
```

```
bool remember
```

```
bool accepted
```

## Signals

```
void wasClosed ()
```

## Private Members

```
Ui::Credentials *ui
```

## Private Slots

```
void on_chkAnonymous_toggled (bool checked)
```

```
void on_pushButton_3_clicked ()
```

```
void on_pushButton_4_clicked ()
```

```
class FeedbackPanel : public QWidget
```

```
    #include <feedbackpanel.h> Abstract base class for a Qt helper widget used in action user feedback.
```

```
    Subclassed by sh::ui::qt::feedbackpanels::Credentials, sh::ui::qt::feedbackpanels::GridForm,  
    sh::ui::qt::feedbackpanels::MsgBox, sh::ui::qt::feedbackpanels::UnixPermissions
```



## Public Functions

```

FeedbackPanel (QWidget *parent = 0)
bool isClosed ()
void waitUntilClosed ()
int preferredHeight ()
void cancelRequested () = 0

```

## Signals

```

void wasClosed ()

```

## Private Members

```

QMutex _mutex
QWaitCondition _closedcondition
bool _closed = false

```

```

class GridForm: public sh::ui::qt::feedbackpanels::FeedbackPanel
    #include <gridform.h> FeedbackPanel for grid based forms.

```

## Public Functions

```

GridForm (QString text, QList<sh::ui::qt::feedbackpanels::GridFormRow> form, QStringList an-
    swers, int defaultanswer = -1, int cancelanswer = -1, QWidget *parent = 0)
~GridForm ()
int preferredHeight ()
void cancelRequested ()
bool isClosed ()
void waitUntilClosed ()

```

## Public Members

```

int answer = -1

```

## Signals

```

void wasClosed ()

```

### Private Members

```
Ui::GridForm *ui
QHash<QPushButton*, int> btn2index
QPushButton *_cancelBtn = 0
QPushButton *_defaultBtn = 0
```

### Private Slots

```
void slot_btnclicked()
```

```
class GridFormInnerSimpleChooser : public QWidget
#include <gridforminnersimplechooser.h> Helper widget for a GridForm.
```

### Public Functions

```
GridFormInnerSimpleChooser (sh::actions::ActionExecutionUserFeedback::Choices
                             *choices, QWidget *parent = 0)
~GridFormInnerSimpleChooser ()
```

### Public Members

```
int answer = -1
QString text
```

### Private Functions

```
bool eventFilter (QObject *obj, QEvent *event)
```

### Private Members

```
QLineEdit *lineedit
sh::actions::ActionExecutionUserFeedback::Choices *choices
QVBoxLayout *mylayout
sh::actions::ActionExecutionUserFeedback::Choice *currentchoice = 0
```

### Private Slots

```
void slot_chosen (int id)
void slot_textedited (QString t)
```

```
class MsgBox : public sh::ui::qt::feedbackpanels::FeedbackPanel
#include <msgbox.h> FeedbackPanel for showing a message, some answer buttons, and optionally a text
input box.
```

## Public Functions

```
MsgBox (QString question, QList<QString> answers, QString icon = "", QString withInputBox =
    QString(), bool inputBoxMultiline = false, int defaultanswer = -1, int cancelanswer = -1,
    int valuePreselectFrom = -1, int valuePreselectTo = -1, QList<QString> answericons =
    QList<QString>(), QWidget *parent = 0)
```

```
~MsgBox ()
```

```
void cancelRequested ()
```

```
bool isClosed ()
```

```
void waitUntilClosed ()
```

## Public Members

```
int answer = -1
```

```
QString text
```

## Signals

```
void wasClosed ()
```

## Private Members

```
Ui::MsgBox *ui
```

```
QHash<QPushButton*, int> btn2index
```

```
QWidget *_initialFocusWidget = 0
```

```
QPushButton *_cancelBtn = 0
```

```
QPushButton *_defaultBtn = 0
```

```
bool _inputBoxMultiline
```

```
int _valuePreselectFrom
```

```
int _valuePreselectTo
```

## Private Slots

```
void slot_btnclicked ()
```

```
class UnixPermissions : public sh::ui::qt::feedbackpanels::FeedbackPanel
    #include <unixpermissions.h> FeedbackPanel for setting one set of unix filesystem permissions.
```

## Public Functions

**UnixPermissions** (QWidget *\*parent* = 0)  
**~UnixPermissions** ()  
void **cancelRequested** ()  
void **setflags** (bool *userMayRead*, bool *userMayWrite*, bool *userMayExecute*, bool *groupMayRead*, bool *groupMayWrite*, bool *groupMayExecute*, bool *othersMayRead*, bool *othersMayWrite*, bool *othersMayExecute*, bool *sticky*, bool *setuid*, bool *setgid*)  
void **setusers** (QStringList *users*, QString *selecteduser*)  
void **setgroups** (QStringList *groups*, QString *selectedgroup*)  
bool **isClosed** ()  
void **waitUntilClosed** ()  
int **preferredHeight** ()

## Public Members

bool **accepted**  
bool **userMayRead**  
bool **userMayWrite**  
bool **userMayExecute**  
bool **groupMayRead**  
bool **groupMayWrite**  
bool **groupMayExecute**  
bool **othersMayRead**  
bool **othersMayWrite**  
bool **othersMayExecute**  
bool **sticky**  
bool **setuid**  
bool **setgid**  
QString **user**  
QString **group**

## Signals

void **wasClosed** ()

## Private Members

*Ui::*UnixPermissions \***ui**

## Private Slots

void **on\_pushButton\_clicked** ()

void **on\_pushButton\_2\_clicked** ()

## 10.2.28 Namespace *sh::ui::web*

**namespace** *sh::ui::web*

The web user interface implementation.

This optional UI provides an application running in a web browser.

## Typedefs

**typedef** QPair<QIcon, int> **WebIconTextBannerItemIcon**

**typedef** QPair<*WebIconTextBannerItemIcon*, QString> **WebIconTextBannerItem**

**typedef** QMap<QString, QString> **WebCommandData**

**class** **WebAboutDialog** : **public** QObject, **public** *sh::ui::web::WebDialog*, **public** *sh::ui::AboutDialog*  
*#include <webaboutdialog.h>* Web based about dialog.

## Public Functions

**WebAboutDialog** ()

QVariant **dialogProperty** (QString *dlgPropertyKey*, QVariant *deflt* = QVariant())  
Returns a dialog property value by key.

void **setDialogProperty** (QString *dlgPropertyKey*, QVariant *value*)  
Sets a dialog property value for a key.

QVariantHash **dialogResult** ()  
Returns the dialog result as hash.

The browser side of a web dialog can ‘accept’ a dialog while closing by setting a non-empty dialog result. This mechanism allows browser side dialogs to return data to the backend. It’s up to the backend to interpret the content of this hash.

If the dialog was cancelled, or is still open, the returned hash is empty. See also `wasClosed()`.

void **triggerDialogEvent** (QString *name*, QJsonObject *data* = QJsonObject())  
Triggers a dialog event (at the browsers).

They in turn typically fetch some data again in order to refresh the ui.

void **setCloseableByUser** (bool v = true)

Sets if this dialog shall be directly closeable by the user by the window decoration. Set to `false` if the user has to react on this dialog by clicking on some controls in the dialog body.

This method must be called during construction.

bool **isCloseableByUser** ()

Returns if this dialog shall be directly closeable by the user by the window decoration.

QJsonObject **toJson** () **override**

Returns the json representation as QJsonObject.

qint64 **dialogId** ()

Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

bool **isInitiated** ()

Returns if this dialog is initialized.

Must be called in main thread.

bool **wasAccepted** ()

Returns if this dialog was 'accepted' by the user (typically by clicking some 'OK' button).

Must be called in main thread.

void **waitClosed** ()

Wait until the user closed the dialog in some way.

May be called in any thread.

void **close** ()

Closes the dialog.

Must be called in main thread.

bool **wasClosed** ()

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \***manager** ()

Returns the *DialogManager* which hosts this dialog.

**class** **WebActionExecutionInfoDialog** : **public** *sh::ui::ActionExecutionInfoDialog*  
*#include <webactionexecutioninfodialog.h>* Web based action execution dialog.

## Public Types

**enum** **MessageBoxButton**

Buttons in a message box from *ActionExecutionUserFeedback*.

*Values:*

**enumerator** **NONE** = 0

**enumerator** **OK** = 1 << 0

**enumerator** **Continue** = 1 << 1

**enumerator** **Cancel** = 1 << 2

```

enumerator Retry = 1 << 3
enumerator Yes = 1 << 4
enumerator No = 1 << 5

```

## Public Functions

**WebActionExecutionInfoDialog** (*sh::actions::ActionExecutionInfo* \*info)

void **setDetails** (QString fv, QString fob, QString tv, QString tob)  
Sets the item details text ('from a/foo.jpg', 'to b/foo.jpg'). .

void **setHead** (QString txt)  
Sets the header text. .

void **setProgress** (bool isDeterminate, quint64 done, quint64 all, QString text)  
Sets the progress. .

int **messageBox** (QString text, QList<QString> answers, QString icon = QString(), int defaultanswer = -1, int cancelanswer = -1, QList<QString> answericons = QList<QString>())

int **inputBox** (QString text, QList<QString> answers, QString \*value, QString icon = QString(), int defaultanswer = -1, int cancelanswer = -1, int valuePreselectFrom = -1, int valuePreselectTo = -1)

int **multilineInputBox** (QString text, QList<QString> answers, QString \*value, QString icon = QString(), int defaultanswer = -1, int cancelanswer = -1)

int **simpleChooserGridform** (QString text, GridformEntries \*entries, QStringList answers, int defaultanswer = -1, int cancelanswer = -1)

bool **credentialsDialog** (QString text, bool showDomain, bool showUsername, bool showPassword, bool showAnonymous, bool showRemember, QString \*domain, QString \*username, QString \*password, bool \*isAnonymous, bool \*isRemember)

bool **unixPermissionsDialog** (bool \*userMayRead, bool \*userMayWrite, bool \*userMayExecute, bool \*groupMayRead, bool \*groupMayWrite, bool \*groupMayExecute, bool \*othersMayRead, bool \*othersMayWrite, bool \*othersMayExecute, bool \*sticky, bool \*setuid, bool \*setgid, QStringList users, QStringList groups, QString \*ownerUser, QString \*ownerGroup)

quint64 **id** ()

quint64 **webts\_created** ()

QString **details\_fromverb** ()

QString **details\_fromobject** ()

QString **details\_toverb** ()

QString **details\_toobject** ()

QString **head** ()

bool **progress\_isDeterminate** ()

quint64 **progress\_done** ()

quint64 **progress\_all** ()

```
QString progress_text ()
QJsonValue userFeedbackAsJsonValue ()
bool isLogicallyVisible ()
    Returns if this dialog is logically visible (i.e. set visible by the action).
void setLogicallyVisible (bool v)
    Sets if this dialog is logically visible (i.e. set visible by the action). .
bool isBackground ()
    Returns if this dialog is currently in background mode (i.e. not visible).
void setBackground (bool v)
    Sets if this dialog is currently in background mode (i.e. not visible). .
void setForceForeground (bool v)
    Sets if this dialog is currently forced to be visible in foreground. .
QString simpleInputDialog (QString text, QString deflt, int valuePreselectFrom = -1, int valuePre-
    selectTo = -1)
int simpleMessageBox (QString text, int buttons = (int)MessageBoxButton::OK, QString icon =
    QString(), int defaultbutton = (MessageBoxButton)0, int cancelbutton =
    (MessageBoxButton)0)
```

### Private Functions

```
void _handleUserFeedback (QString kind, std::shared_ptr<UserFeedback> userfeedback)
void _triggerChanged ()
```

### Private Members

```
qint64 _id
QString _details_fv
QString _details_fob
QString _details_tv
QString _details_tob
QString _head
bool _progress_isDeterminate
quint64 _progress_done
quint64 _progress_all
QString _progress_text
std::shared_ptr<UserFeedback> _currentUserFeedback = 0
QMutex _currentUserFeedbackMutex
QWaitCondition _currentUserFeedbackChangedCondition
qint64 _webts_created
```



## Private Static Functions

```
QByteArray iconToBase64Src (QString icon, int sizeInPt)
QList<QVariant> iconsToBase64Srcs (QStringList icons, int sizeInPt)
```

## Friends

```
friend class WebActionManager
class UserFeedback : public QMap<QString, QVariant>
    #include <webactionexecutioninfodialog.h>
```

## Public Members

```
qint64 id = -1
bool answered = false
class WebActionExecutionInfoPanel : public sh::ui::ActionExecutionInfoPanel
    #include <webactionexecutioninfopanel.h> Web based action execution info panel.
```

## Public Functions

```
WebActionExecutionInfoPanel (sh::actions::ActionExecutionInfo *info, QColor color)
~WebActionExecutionInfoPanel ()
void setLabel (QString s)
    Sets the label text. .
void setProgress (bool isProgressDeterminate, quint64 progressDone, quint64 progressAll)
    Sets the progress. .
void setForceForeground (bool v)
    Sets if the associated action is currently forced to be visible in foreground. .
void setPanelVisible (bool v)
    Sets if the panel is visible. .
void setWidth (int width)
    Sets the panel with (in pixels). .
qint64 id ()
QString label ()
bool isProgressDeterminate ()
quint64 progressDone ()
quint64 progressAll ()
bool forceForeground ()
bool panelVisible ()
int width ()
QColor color ()
sh::actions::ActionExecutionInfo *info ()
```

```
void onClicked (std::function<void>
    > fcnQObject *owner = 0)Sets a handler for a click on the button (optionally bound to an owner lifetime).

void onDestroyed (std::function<void>
    > fcnQObject *owner = 0)Sets a handler for panel removal (optionally bound to an owner lifetime).

void onVisibilityChanged (std::function<void>
    > fcnQObject *owner = 0)Sets a handler for panel visibility changes (optionally bound to an owner lifetime).
```

## Private Functions

```
void _triggerChanged ()
```

## Private Members

```
QColor _color
sh::actions::ActionExecutionInfo *_info
qint64 _id
QString _label
bool _isProgressDeterminate
quint64 _progressDone
quint64 _progressAll
bool _forceForeground
bool _panelVisible
int _width
```

## Friends

```
friend class WebActionManager

class WebActionManager : public QObject, public sh::ui::web::WebModule
    #include <webactionmanager.h> Manages sh::actions::AbstractActionItem handling, e.g.
    showing them in the toolbar, executing them, and displaying their user interface.
```

## Public Functions

```
WebActionManager ()
```

```
void initialize ()
```

```
std::shared_ptr<WebActionExecutionInfoPanel> addInfoPanel (int position,
    sh::actions::ActionExecutionInfo
    *info, QColor color = QColor())
```

Adds and returns a new action info panel.

```
std::shared_ptr<WebActionExecutionInfoDialog> addActionExecutionInfoDialog (sh::actions::ActionExecutionInfo
    *info)
```

Adds and returns a new action info dialog.

```
void refreshToolBar (std::shared_ptr<sh::actions::ActionInstantiation> instantiation)
    Refreshes the main toolbar.

bool rootCommand (WebServerEngineRequest *request, QString webadapter)
    Returns the root ('index.html') content. .

void setEngine (WebServerEngine *webserver)
    Assigns this web module to a sh::ui::web::WebServerEngine. .

WebServerEngine *webserver ()
    Returns the sh::ui::web::WebServerEngine this web module is assigned to (may be nullptr).
```

## Public Static Functions

```
bool executeCommand_byQObjectReflection (QObject *o, WebServerEngineRequest *re-
                                         quest, QString command, QString justLeft-
                                         side = QString())

    Convenience function often used by executeCommand().

    For a QObject, it searches there for a slot with the name cmd_{command}, with ``'s
    replaced by '_s (so, for the command "foo/getBars", it searches for
    slotcmd_foo_getBars`). If it finds one, it gets executed (in a worker thread). The request is
    considered as handled, if so.

    If there is such a slot, but with appended __M in name, it is called in main thread!
Return If it handled (answered) the command.
```

## Private Functions

```
bool executeCommand (WebServerEngineRequest *request, QString command) override
    Executes command as requested by the browser, if responsible.

    Read parameters, check if this module is responsible for answering, and then write an answer with
    request.

    Implementations often use executeCommand_byQObjectReflection() for convenience.
Return If it handled (answered) the command.

void _observeToolBarAction (sh::actions::AbstractActionItem *a)

std::shared_ptr<sh::actions::AbstractActionItem> resolveActionpath (QString      saction-
                                                                    path,      QStringList
                                                                    *actionpath,
                                                                    QList<QPair<QString,
                                                                    std::shared_ptr<sh::actions::AbstractActionItem>
                                                                    *subactions>

void _triggerActionuiChanged ()
```

### Private Members

```
QList<WebActionExecutionInfoPanel*> _infopanelss
QList<WebActionExecutionInfoDialog*> _infodialogs
QTimer _triggerActionuiChangedTimer
QTimer _triggerToolbarRefreshTimer
QMap<QString, std::shared_ptr<sh::actions::SubmenuActionItem>> _permanentToolbarActions
```

### Private Slots

```
void cmd__answer_userfeedback__M(WebServerEngineRequest *request)
void cmd__execute(WebServerEngineRequest *request)
void cmd__get(WebServerEngineRequest *request)
void cmd__get_ui__M(WebServerEngineRequest *request)
void cmd__icon(WebServerEngineRequest *request)
void cmd__panel_clicked__M(WebServerEngineRequest *request)
```

### Private Static Attributes

```
QRegularExpression reActionTextAccel
```

### Friends

```
friend class WebActionExecutionInfoPanel
friend class WebActionExecutionInfoDialog
class WebDetailPanelManager : public QObject, public sh::ui::web::WebModule
    #include <webdetailpanelmanager.h> Manages the detail panel (typically in bottom part of main window).
```

### Public Functions

```
WebDetailPanelManager()

void load(QList<std::shared_ptr<sh::filesystem::FilesystemNode>> nodes)
    Loads details for a given list of nodes (typically called after they get selected).

bool rootCommand(WebServerEngineRequest *request, QString webadapter)
    Returns the root ('index.html') content. .

void setEngine(WebServerEngine *webserver)
    Assigns this web module to a sh::ui::web::WebServerEngine. .

WebServerEngine *webserver()
    Returns the sh::ui::web::WebServerEngine this web module is assigned to (may be nullptr).
```

## Public Static Functions

bool **executeCommand\_byQObjectReflection** (QObject \*o, *WebServerEngineRequest* \*request, QString command, QString justLeftSide = QString())

Convenience function often used by *executeCommand()*.

For a QObject, it searches there for a slot with the name cmd\_{command}, with `/'s replaced by '\_s' (so, for the command "foo/getBars", it searches for slotcmd\_foo\_getBars). If it finds one, it gets executed (in a worker thread). The request is considered as handled, if so.

If there is such a slot, but with appended \_\_M in name, it is called in main thread!

**Return** If it handled (answered) the command.

## Private Functions

bool **executeCommand** (*WebServerEngineRequest* \*request, QString command) **override**

Executes command as requested by the browser, if responsible.

Read parameters, check if this module is responsible for answering, and then write an answer with request.

Implementations often use *executeCommand\_byQObjectReflection()* for convenience.

**Return** If it handled (answered) the command.

## Private Members

QMutex **\_mutex**

QList<std::shared\_ptr<*sh::panelDetails::PanelDetail*>> **\_panelDetails**

qint64 **\_webts\_lastupdate**

## Private Slots

void **cmd\_\_get\_details** (*WebServerEngineRequest* \*request)

void **cmd\_\_get\_thumbnail** (*WebServerEngineRequest* \*request)

void **cmd\_\_link\_clicked** (*WebServerEngineRequest* \*request)

**class WebDialog**: public *sh::tools::Jsonable*

*#include <webdialog.h>* Abstract base class for a web based dialog. Typically used for also implementing some *sh::ui::Dialog*.

A web dialog is a dialog window presented to the user on browser side. A web dialog implementation consists of a backend part (a subclass of *WebDialog*) and a browser side part (basically a subclass of *shwebui.Dialog*).

For implementing a web dialog, implement a subclass of *WebDialog* (which in turn will specify the browser side part) and *sh::ui::Dialog*. Create such dialogs by means of *WebDialogManager*.

Communicating values between backend and browser typically works by *dialogProperty()* and *setDialogProperty()*, also *dialogResult()* for the dialog result after closing.

Subclassed by *sh::ui::web::WebAboutDialog*, *sh::ui::web::WebExceptionDialog*,  
*sh::ui::web::WebFilePropertyDialog*, *sh::ui::web::WebLogViewDialog*,

*sh::ui::web::WebManageBookmarksDialog,* *sh::ui::web::WebManageProfilesDialog,*  
*sh::ui::web::WebOpenWithDialog, sh::ui::web::WebStoreProfileDialog, sh::ui::web::WebTuningDialog*

## Public Functions

**WebDialog** (QString *dialogClassName*)

See *WebDialogManager*.

### Parameters

- **dialogClassName**: The name of the shwebui.Dialog subclass which implements the web dialog on browser side.

QVariant **dialogProperty** (QString *dlgPropertyKey*, QVariant *deflt* = QVariant())

Returns a dialog property value by key.

void **setDialogProperty** (QString *dlgPropertyKey*, QVariant *value*)

Sets a dialog property value for a key.

QVariantHash **dialogResult** ()

Returns the dialog result as hash.

The browser side of a web dialog can ‘accept’ a dialog while closing by setting a non-empty dialog result. This mechanism allows browser side dialogs to return data to the backend. It’s up to the backend to interpret the content of this hash.

If the dialog was cancelled, or is still open, the returned hash is empty. See also *wasClosed()*.

void **triggerDialogEvent** (QString *name*, QJsonObject *data* = QJsonObject())

Triggers a dialog event (at the browsers).

They in turn typically fetch some data again in order to refresh the ui.

void **setCloseableByUser** (bool *v* = true)

Sets if this dialog shall be directly closeable by the user by the window decoration. Set to *false* if the user has to react on this dialog by clicking on some controls in the dialog body.

This method must be called during construction.

bool **isCloseableByUser** ()

Returns if this dialog shall be directly closeable by the user by the window decoration.

**~WebDialog** ()

QJsonObject **toJson** () **override**

Returns the json representation as QJsonObject.

## Private Members

QVariantHash **\_properties**

QString **\_dialogClassName**

qint64 **\_webts\_created**

## Friends

**friend class** WebDialogManager

**class** WebDialogManager : public QObject, public *sh::ui::web::WebModule*, public *sh::ui::DialogManager*  
*#include <webdialog.h>* A *DialogManager* used in Web ui.

## Public Functions

**WebDialogManager** ()

*WebServerEngine* \***webserver** ()

Return the *WebServerEngine* hosting this dialog.

bool **rootCommand** (*WebServerEngineRequest* \*request, QString webadapter)

Returns the root ('index.html') content. .

void **setEngine** (*WebServerEngine* \*webserver)

Assigns this web module to a *sh::ui::web::WebServerEngine*. .

std::shared\_ptr<*Dialog*> **getDialogById** (qint64 id)

Returns a *Dialog* by dialog id.

Must be called in main thread.

QList<std::shared\_ptr<*Dialog*>> **getAllDialogs** ()

Returns a list of all dialogs currently shown (in order of creation).

Must be called in main thread.

QList<qint64> **getAllDialogIds** ()

Returns a list of dialog ids of all dialogs currently shown (in order of creation).

Must be called in main thread.

template<class **TDlg**, typename ...**Args**>

std::shared\_ptr<*TDlg*> **createAndShowDialog** (*Args...* args)

Creates and shows a *Dialog* by class and constructor parameters.

Those dialogs (TDlg) have to implement *Dialog* (typically a subclass of it, representing a particular dialog, like *FilePropertyDialog*), and also some ui mode (e.g. qt, web) specific class (depends on that ui mode).

You typically should not have to use it outside of *MainWindow* or subclasses. The *MainWindow* interface allows to create all available dialogs in a ui mode independent way.

May be called in any thread.

**Return** The created *Dialog* instance.

## Public Static Functions

bool **executeCommand\_byQObjectReflection** (QObject \*o, *WebServerEngineRequest* \*request, QString command, QString justLeft-side = QString())

Convenience function often used by *executeCommand()*.

For a QObject, it searches there for a slot with the name cmd\_{command}, with `/'s replaced by '\_'s (so, for the command "foo/get\_bars", it searches for slotcmd\_foo\_get\_bars). If it finds one, it gets executed (in a worker thread). The request is considered as handled, if so.

If there is such a slot, but with appended `__M` in name, it is called in main thread!

**Return** If it handled (answered) the command.

## Private Functions

`std::shared_ptr<Dialog> getDialogForRequest (WebServerEngineRequest *request)`

Returns the *WebDialog* referred to the request (i.e. its `dialogId` parameter).

`bool executeCommand (WebServerEngineRequest *request, QString command) override`

Executes `command` as requested by the browser, if responsible.

Read parameters, check if this module is responsible for answering, and then write an answer with request.

Implementations often use *executeCommand\_byQObjectReflection()* for convenience.

**Return** If it handled (answered) the command.

`void showDialog (std::shared_ptr<Dialog> dialog) override`

This method implements the ui mode specific mechanism for showing a dialog.

Must be called in main thread.

## Private Slots

`void cmd__change_dialog_property__M (WebServerEngineRequest *request)`

`void cmd__closed_in_browser__M (WebServerEngineRequest *request)`

`void cmd__list__M (WebServerEngineRequest *request)`

`class WebExceptionDialog: public QObject, public sh::ui::web::WebDialog, public sh::ui::ExceptionDialog  
#include <webexceptiondialog.h> Web based exception dialog.`

## Public Functions

`WebExceptionDialog (QString error1, QString error2, QString details, QString icon, bool  
mayRetry, bool mayClose, bool mayCancel, bool showLoglabelAndDe-  
tails)`

`ExceptionDialogResult answer () override`

Returns the answer the user has given in this dialog.

`QVariant dialogProperty (QString dlgPropertyKey, QVariant deflt = QVariant())`

Returns a dialog property value by key.

`void setDialogProperty (QString dlgPropertyKey, QVariant value)`

Sets a dialog property value for a key.

`QVariantHash dialogResult ()`

Returns the dialog result as hash.

The browser side of a web dialog can ‘accept’ a dialog while closing by setting a non-empty dialog result. This mechanism allows browser side dialogs to return data to the backend. It’s up to the backend to interpret the content of this hash.

If the dialog was cancelled, or is still open, the returned hash is empty. See also `wasClosed()`.



void **triggerDialogEvent** (QString *name*, QJsonObject *data* = QJsonObject())

Triggers a dialog event (at the browsers).

They in turn typically fetch some data again in order to refresh the ui.

void **setCloseableByUser** (bool *v* = true)

Sets if this dialog shall be directly closeable by the user by the window decoration. Set to `false` if the user has to react on this dialog by clicking on some controls in the dialog body.

This method must be called during construction.

bool **isCloseableByUser** ()

Returns if this dialog shall be directly closeable by the user by the window decoration.

QJsonObject **toJson** () **override**

Returns the json representation as QJsonObject.

QString **error1** ()

Returns the 1st error text.

QString **error2** ()

Returns the 2nd error text.

QString **details** ()

Returns the error details.

QString **icon** ()

Returns the icon (as name resolveable by `sh::base::IconManager`).

bool **mayRetry** ()

Returns if it's allowed to retry.

bool **mayClose** ()

Returns if it's allowed to just close and continue without closing Shallot.

bool **mayCancel** ()

Returns if it's allowed to cancel, i.e. throwing a `sh::exceptions::CancelException`.

bool **showLoglabelAndDetails** ()

Returns if the dialog shall allow to read the details.

qint64 **dialogId** ()

Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

bool **isInitied** ()

Returns if this dialog is initialized.

Must be called in main thread.

bool **wasAccepted** ()

Returns if this dialog was 'accepted' by the user (typically by clicking some 'OK' button).

Must be called in main thread.

void **waitClosed** ()

Wait until the user closed the dialog in some way.

May be called in any thread.

void **close** ()

Closes the dialog.

Must be called in main thread.

bool **wasClosed** ()

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \***manager** ()

Returns the *DialogManager* which hosts this dialog.

## Private Slots

void **cmd\_\_icon\_\_M** (*WebServerEngineRequest* \*request)

**class WebFilePropertyDialog** : public QObject, public *sh::ui::web::WebDialog*, public *sh::ui::FilePropertyDialog*  
#include <webfilepropertydialog.h> Web based file property dialog.

## Public Functions

void **refresh** () **override**

Refreshes the dialog content.

*FilePropertyDialogTabTableView* \***createTabViewTable** () **override**

Creates a new tab view table sub-widget.

*FilePropertyDialogTabTextView* \***createTabViewText** () **override**

Creates a new tab view text sub-widget.

*FilePropertyDialogTabIconTextBannerView* \***createTabViewIconTextBanner** () **override**

Creates a new tab view icon text banner sub-widget.

**WebFilePropertyDialog** (QList<std::shared\_ptr<*sh::filesystem::FilesystemNode*>> nodes)

**~WebFilePropertyDialog** ()

QVariant **dialogProperty** (QString *dlgPropertyKey*, QVariant *deflt* = QVariant())

Returns a dialog property value by key.

void **setDialogProperty** (QString *dlgPropertyKey*, QVariant *value*)

Sets a dialog property value for a key.

QVariantHash **dialogResult** ()

Returns the dialog result as hash.

The browser side of a web dialog can ‘accept’ a dialog while closing by setting a non-empty dialog result. This mechanism allows browser side dialogs to return data to the backend. It’s up to the backend to interpret the content of this hash.

If the dialog was cancelled, or is still open, the returned hash is empty. See also wasClosed().

void **triggerDialogEvent** (QString *name*, QJsonObject *data* = QJsonObject())

Triggers a dialog event (at the browsers).

They in turn typically fetch some data again in order to refresh the ui.

void **setCloseableByUser** (bool *v* = true)

Sets if this dialog shall be directly closeable by the user by the window decoration. Set to *false* if the user has to react on this dialog by clicking on some controls in the dialog body.

This method must be called during construction.

bool **isCloseableByUser** ()

Returns if this dialog shall be directly closeable by the user by the window decoration.

QJsonObject **toJson** () **override**

Returns the json representation as QJsonObject.

QList<std::shared\_ptr<*FilePropertyDialogTab*>> **tabs** ()

Returns a list of all tabs.

*sh::ui::FilePropertyDialogTabActionsView* \***widgetAt** (std::shared\_ptr<*FilePropertyDialogTab*>  
tab, int i)

Returns the widget from a tab at a certain index.

int **widgetCount** (std::shared\_ptr<*FilePropertyDialogTab*> tab)

Returns the number of widgets in a tab.

qint64 **dialogId** ()

Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

bool **isInit** ()

Returns if this dialog is initialized.

Must be called in main thread.

bool **wasAccepted** ()

Returns if this dialog was 'accepted' by the user (typically by clicking some 'OK' button).

Must be called in main thread.

void **waitClosed** ()

Wait until the user closed the dialog in some way.

May be called in any thread.

void **close** ()

Closes the dialog.

Must be called in main thread.

bool **wasClosed** ()

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \***manager** ()

Returns the *DialogManager* which hosts this dialog.

## Public Static Functions

void **addTabFactory** (int i, std::shared\_ptr<*FilePropertyDialogTabFactory*> factory)

Adds a *FilePropertyDialogTabFactory* so the Properties dialogs show its content.

See also the REGISTER\_FILEPROPERTYDIALOGTAB macro.

### Parameters

- **i**: An index which controls the display order. Use one of the REGISTER\_FILEPROPERTYDIALOGTAB\_INDEX\_\* values in *FilePropertyDialogTabFactory* as base values.

### Private Functions

```
void tabwidgetTableSelect (int cookie, int tabidx, int widgetidx, QList<int> lsel)  
void triggerAction (int cookie, int tabidx, int widgetidx, int btnidx)  
QJsonArray getProperties ()
```

### Private Members

```
int _cookie = 0  
QTimer _refreshTimer  
QList<WebFilePropertyDialogChild> _childs
```

### Private Slots

```
void cmd__get_properties__M (WebServerEngineRequest *request)  
void cmd__refresh__M (WebServerEngineRequest *request)  
void cmd__trigger_action__M (WebServerEngineRequest *request)  
void cmd__tabwidget_table_select__M (WebServerEngineRequest *request)
```

### class *WebFilePropertyDialogChild*

Subclassed by *sh::ui::web::WebFilePropertyDialog::WebFilePropertyDialogTabActionsView*,  
*sh::ui::web::WebFilePropertyDialog::WebFilePropertyDialogTabIconTextBannerView*,  
*sh::ui::web::WebFilePropertyDialog::WebFilePropertyDialogTabTableView*,  
*sh::ui::web::WebFilePropertyDialog::WebFilePropertyDialogTabTextView*

### Public Functions

```
WebFilePropertyDialogChild (WebFilePropertyDialog *dlg)  
~WebFilePropertyDialogChild ()  
class WebFilePropertyDialogTabActionsView: public sh::ui::FilePropertyDialogTabActionsView, public
```

### Public Functions

```
WebFilePropertyDialogTabActionsView (WebFilePropertyDialog *dlg)  
void setVisible (bool v) override  
    Sets the view visible or hidden. .  
void setContent (FilePropertyDialogTabViewContent *cnt) override  
    Sets the main widget. .  
void setButtons (QList<QString> buttons) override  
    Sets the list of buttons. .  
void setLabelText (QString text)  
QJsonObject toJson () override  
    Returns the json representation as QJsonObject.
```

*FilePropertyDialogTabViewContent* \***content** ()

The main widget.

QList<QString> **buttons** ()

The list of buttons.

void **onButtonTriggered** (std::function<void> int i

> *fcn*, QObject \**owner* = 0) Sets a handler for a click on one of the buttons (optionally bound to an owner lifetime).

## Private Members

bool **\_visible** = false

QString **\_labelText**

## Friends

**friend class** WebFilePropertyDialog

**class** WebFilePropertyDialogTabIconTextBannerView : **public** *sh::ui::FilePropertyDialogTabIconTextBannerView*

## Public Functions

**WebFilePropertyDialogTabIconTextBannerView** (*WebFilePropertyDialog* \**dlg*)

void **clear** () **override**

Clears the contents.

void **insertIcon** (QIcon *icon*, int *sizePt* = 24) **override**

Adds an icon.

void **insertText** (QString *text*) **override**

Adds a text.

QJsonObject **toJson** () **override**

Returns the json representation as QJsonObject.

## Private Members

QList<*WebIconTextBannerItem*> **\_content**

**class** WebFilePropertyDialogTabTableView : **public** *sh::ui::FilePropertyDialogTabTableView*, **public** *sh::ui::WebFilePropertyDialogTabTableView*

## Public Types

**typedef** QPair<int, int> **ItemIndex**

### Public Functions

**WebFilePropertyDialogTabTableView** (*WebFilePropertyDialog \*dlg*)

void **setContent** (QList<QPair<QString, QString>> *content*) **override**  
Sets the content.

QList<*ItemIndex*> **getSelectedItems** () **override**  
Returns the selected cells.

QVariant **getItemValue** (int *r*, int *c*) **override**  
Returns a value of a cell.

QJsonObject **toJson** () **override**  
Returns the json representation as QJsonObject.

void **setSelectedItems** (QList<int> *selitms*)

### Private Members

QList<QPair<QString, QString>> **\_content**

QList<int> **\_selectedItems**

**class WebFilePropertyDialogTabTextView**: public *sh::ui::FilePropertyDialogTabTextView*, public *sh::to*

### Public Functions

**WebFilePropertyDialogTabTextView** (*WebFilePropertyDialog \*dlg*)

void **setContent** (QString *content*) **override**  
Sets the content.

QJsonObject **toJson** () **override**  
Returns the json representation as QJsonObject.

### Private Members

QString **\_content**

**class WebFileView**: public *sh::ui::FileView*  
*#include <webfileview.h>* Web based file view.

### Public Functions

**WebFileView** (*WebFileViewManager \*manager*)

*sh::ui::ColumnDimensions* \***columnDimensions** () **override**  
Returns the column dimensions handler. .

*sh::tools::HistoryTracker*<std::shared\_ptr<const *sh::filesystem::Eurl*>> \***historyTracker** () **override**  
Returns the history tracker. .

*FileViewMode* **viewmode** () **override**  
Returns the view mode (icons, list, ... ?). .

```

void setViewmode (FileViewMode m) override
    Sets the view mode. .

int index () override
    Returns the position of this file view within the panel. .

void gotoDir (std::shared_ptr<sh::filesystem::FilesystemNode> n) override
    Jumps to a new current directory.

    Implementations have to call the base class implementation inside.

sh::filesystem::SizeFormatting getSizeFormattingMode () override
    Returns the mode how file sizes are formatted for displaying. .

void setSizeFormattingMode (sh::filesystem::SizeFormatting mode) override
    Sets the mode how file sizes are formatted for displaying. .

bool hiddenFilesVisible () override
    Returns if hidden files are visible. .

void setHiddenFilesVisible (bool v) override
    Sets the visibility of hidden files. .

int iconDimension () override
    Returns the icon size (in pixels). .

void setIconDimension (int v) override
    Sets the icon size (in pixels). .

void setSort (int column, Qt::SortOrder order) override
    Sets how to sort this view. .

int sortColumn () override
    Returns the index of the current sort column. .

Qt::SortOrder sortOrder () override
    Returns the current sort order. .

QList<std::shared_ptr<sh::filesystem::FilesystemNode>> selectedNodes () override
    Returns the nodes which the user has selected in this fileview. .

void setSelection (const QList<std::shared_ptr<sh::filesystem::FilesystemNode>> nodes)
    override
    Sets the node selection of this fileview. .

QList<std::shared_ptr<sh::filesystem::FilesystemNode>> getAllVisibleNodes () override
    Returns a list of all nodes currently listed in this file view. .

qint64 id ()
    Returns the unique identifier.

std::shared_ptr<sh::filesystem::FilesystemModelFileviewProxy> model ()
    Returns the filesystem model bound to this view.

std::shared_ptr<sh::filesystem::FilesystemNode> node ()
    Returns the current directory.

void reload (bool skipModel = false)
    Reloads the data inside this file view.

sh::filesystem::FilesystemModelFileviewProxy *filemodel ()
    Returns the filesystem model for this view. .

void setFilemodel (sh::filesystem::FilesystemModelFileviewProxy *model)
    Sets the filesystem model for this view. .

```

## Signals

void **currentNodeChanged** ()  
void **modelChanged** ()  
void **selectionChanged** ()  
    Triggered when the selection have changed.  
void **viewOptionChanged** ()  
    Triggered when some view options have changed.

## Private Functions

void **\_set\_model** ()  
void **setSelectedNode** (std::shared\_ptr<sh::filesystem::FilesystemNode> node)  
void **setCheckedNodes** (QList<std::shared\_ptr<sh::filesystem::FilesystemNode>> nodes)

## Private Members

std::shared\_ptr<sh::filesystem::FilesystemModelFileviewProxy> **\_model**  
qint64 **\_id**  
*FileViewMode* **\_viewmode** = *FileViewMode::ListView*  
*sh::filesystem::SizeFormatting* **\_sizeformatting** = *sh::filesystem::SizeFormatting::SizeFormattingModePrefixes*  
bool **\_hiddenFilesVisible** = false  
int **\_iconDimension** = 20  
int **\_sortColumn** = 0  
Qt::SortOrder **\_sortOrder** = Qt::SortOrder::AscendingOrder  
std::shared\_ptr<sh::filesystem::FilesystemNode> **\_selectedNode**  
QList<std::shared\_ptr<sh::filesystem::FilesystemNode>> **\_checkedNodes**  
*WebFileViewManager* \* **\_manager**

## Friends

**friend class** WebFileViewManager  
**class WebFileViewManager** : public QObject, public *sh::ui::web::WebModule*  
    #include <webfileview.h> Manages *WebFileView* instances.



## Public Functions

bool **executeCommand** (*WebServerEngineRequest* \*request, QString command) **override**

Executes command as requested by the browser, if responsible.

Read parameters, check if this module is responsible for answering, and then write an answer with request.

Implementations often use *executeCommand\_byQObjectReflection()* for convenience.

**Return** If it handled (answered) the command.

std::shared\_ptr<*WebFileView*> **addFileView** (bool reinitialize)

void **setCurrentFileViewByIndex** (int idx)

*sh::ui::web::WebFileView* \***currentFileView** ()

int **fileViewCount** ()

void **removeFileView** (int i)

*sh::ui::web::WebFileView* \***getFileView** (int i)

int **getFileViewIndex** (*sh::ui::FileView* \*fileview)

bool **rootCommand** (*WebServerEngineRequest* \*request, QString webadapter)

Returns the root ('index.html') content. .

void **setEngine** (*WebServerEngine* \*webserver)

Assigns this web module to a *sh::ui::web::WebServerEngine*. .

*WebServerEngine* \***webserver** ()

Returns the *sh::ui::web::WebServerEngine* this web module is assigned to (may be nullptr).

## Signals

void **activeSelectionChanged** ()

void **fileViewOptionsChanged** (int i)

void **fileViewCountChanged** ()

void **currentFileViewChanged** ()

## Public Static Functions

bool **executeCommand\_byQObjectReflection** (QObject \*o, *WebServerEngineRequest* \*request, QString command, QString justLeft-side = QString())

Convenience function often used by *executeCommand()*.

For a QObject, it searches there for a slot with the name cmd\_{command}, with `/'s replaced by '\_'s (so, for the command "foo/getBars", it searches for slotcmd\_foo\_getBars). If it finds one, it gets executed (in a worker thread). The request is considered as handled, if so.

If there is such a slot, but with appended \_\_M in name, it is called in main thread!

**Return** If it handled (answered) the command.

### Private Members

```
QList<std::shared_ptr<WebFileView>> _fileviews
int _icurrentfileview = 0
```

### Private Slots

```
void cmd__checkednodes_changed__M(WebServerEngineRequest *request)
void cmd__focus_current__M(WebServerEngineRequest *request)
void cmd__get__M(WebServerEngineRequest *request)
void cmd__list_view_items__M(WebServerEngineRequest *request)
void cmd__node_activated__M(WebServerEngineRequest *request)
void cmd__selection_changed__M(WebServerEngineRequest *request)
```

```
class WebI18n
#include <webi18n.h> Web related helpers for internationalization/translations.
```

### Public Static Functions

```
QString get ()
Returns the Javascript representation for translated strings (in current ui language).
```

```
class WebLogViewDialog : public QObject, public sh::ui::web::WebDialog, public sh::ui::LogViewDialog
#include <weblogviewdialog.h> Web based log view dialog.
```

### Public Functions

```
WebLogViewDialog (QString headtext, QString subheadtxt, sh::base::LogSeverity minseverity)
```

```
QVariant dialogProperty (QString dlgPropertyKey, QVariant deflt = QVariant())
Returns a dialog property value by key.
```

```
void setDialogProperty (QString dlgPropertyKey, QVariant value)
Sets a dialog property value for a key.
```

```
QVariantHash dialogResult ()
Returns the dialog result as hash.
```

The browser side of a web dialog can ‘accept’ a dialog while closing by setting a non-empty dialog result. This mechanism allows browser side dialogs to return data to the backend. It’s up to the backend to interpret the content of this hash.

If the dialog was cancelled, or is still open, the returned hash is empty. See also wasClosed().

```
void triggerDialogEvent (QString name, QJsonObject data = QJsonObject())
Triggers a dialog event (at the browsers).
```

They in turn typically fetch some data again in order to refresh the ui.

```
void setCloseableByUser (bool v = true)
Sets if this dialog shall be directly closeable by the user by the window decoration. Set to false if the user has to react on this dialog by clicking on some controls in the dialog body.
```

This method must be called during construction.

**bool isCloseableByUser ()**  
Returns if this dialog shall be directly closeable by the user by the window decoration.

**QJsonObject toJson () override**  
Returns the json representation as QJsonObject.

**QString headtext ()**  
Returns the header text.

**QString subheadtext ()**  
Returns the 2nd header text.

**sh::base::LogSeverity minimumSeverity ()**  
Returns the minimum severity.

**qint64 dialogId ()**  
Returns the dialog id.  
  
Each instance has an id unique in the complete Shallot process lifetime.  
  
Must be called in main thread.

**bool isInitiated ()**  
Returns if this dialog is initialized.  
  
Must be called in main thread.

**bool wasAccepted ()**  
Returns if this dialog was 'accepted' by the user (typically by clicking some 'OK' button).  
  
Must be called in main thread.

**void waitClosed ()**  
Wait until the user closed the dialog in some way.  
  
May be called in any thread.

**void close ()**  
Closes the dialog.  
  
Must be called in main thread.

**bool wasClosed ()**  
Returns if this dialog was closed.  
  
Must be called in main thread.

**DialogManager \*manager ()**  
Returns the *DialogManager* which hosts this dialog.

## Private Slots

**void cmd\_\_get\_messages (WebServerEngineRequest \*request)**

**class WebMainWindow: public QObject, public sh::ui::MainWindow, public sh::ui::web::WebModule**  
*#include <webmainwindow.h>* The web main window implementation.

## Public Functions

**WebMainWindow** (QString *dispatcherUrl*, QByteArray *wtoken*)

**~WebMainWindow** ()

*WebServerEngine* \***webserver** ()

Returns the *WebServerEngine* instance for this main window.

void **initialize** () **override**

Initializes this main window. .

std::shared\_ptr<*DialogManager*> **dialogManager** () **override**

Returns the *DialogManager* which creates and drives *Dialogs* for this main window.

You typically should not need to use it directly.

int **fileViewCount** () **override**

Returns the current number of file views. .

void **addFileView** (bool *reinitialize* = false) **override**

Adds a new file view. .

void **removeFileView** (int *i*) **override**

Removes a file view. .

*sh::ui::FileView* \***currentFileView** () **override**

Returns the file view which is currently active (i.e. focussed). .

*sh::ui::FileView* \***getFileView** (int *i*) **override**

Returns a file view by position index. .

void **jumpToNode** (std::shared\_ptr<*sh::filesystem::FilesystemNode*> *node*) **override**

Let the current view jump to another location. .

std::shared\_ptr<*sh::filesystem::FilesystemNode*> **currentDirectory** () **override**

Gets the *sh::filesystem::FilesystemNode* selected in the current view. .

void **setTitlePattern** (QString *value*) **override**

Sets the window title pattern. .

void **setTreeVisible** (bool *v*) **override**

Sets the visibility of the directory tree. .

void **setTreeSticky** (bool *v*) **override**

Sets if the directory tree should follow the active file view. .

void **setFileDetailsPanelVisible** (bool *v*) **override**

Sets the visibility of the file details panel. .

std::shared\_ptr<*sh::ui::ActionExecutionInfoPanel*> **addInfoPanel** (int *position*,  
*sh::actions::ActionExecutionInfo*  
\**info*, QColor *color* =  
QColor()) **override**

Creates a new action execution panel.

Let this smart pointer die for removing it.

void **jumpbarSetTextMode** () **override**

Sets the jumpbar to text input mode. .

void **jumpbarSetButtonMode** () **override**

Sets the jumpbar to button mode. .

```

bool jumpbarIsTextMode () override
    Returns if the jumpbar is in text input mode. .

QString toolbarPosition () override
    Returns the current toolbar position. .

void setToolbarPosition (QString pos) override
    Sets the toolbar position. .

QString detailPanelPosition () override
    Returns the current detail panel position. .

void setDetailPanelPosition (QString pos) override
    Sets the detail panel position. .

std::shared_ptr<AboutDialog> showAboutDialog () override
    Shows and returns an 'About' dialog. .

std::shared_ptr<ManageProfilesDialog> showManageProfilesDialog () override
    Shows and returns a 'Manage Saved Settings' dialog. .

std::shared_ptr<OpenWithDialog> showOpenWithDialog (std::shared_ptr<sh::filesystem::FilesystemNode>
                                                    node,
                                                    QList<std::shared_ptr<sh::tools::filetypes::OpenMethod>>
                                                    openMethods) override

    Shows and returns a 'Open With' dialog. .

std::shared_ptr<StoreProfileDialog> showStoreProfileDialog (std::shared_ptr<const
                                                            sh::filesystem::Eurl> eurl)
                                                            override

    Shows and returns a 'Save Settings' dialog. .

std::shared_ptr<TuningDialog> showTuningDialog () override
    Shows and returns a 'Tuning' dialog. .

std::shared_ptr<LogViewDialog> showLogViewDialog (QString headtext = QString(),
                                                    QString subheadtxt = QString(),
                                                    sh::base::LogSeverity minseverity =
                                                    sh::base::LogSeverity::_DEFAULT)
                                                    override

    Shows and returns a 'Log' dialog. .

std::shared_ptr<ManageBookmarksDialog> showManageBookmarksDialog () override
    Shows and returns a 'Manage Bookmarks' dialog. .

std::shared_ptr<FilePropertyDialog> showFilePropertyDialog (QList<std::shared_ptr<sh::filesystem::FilesystemNode>
                                                            nodes) override

    Shows and returns a 'File Properties' dialog. .

std::shared_ptr<ActionExecutionInfoDialog> createActionExecutionInfoDialog (sh::actions::ActionExecutionInfo
                                                                            *info)
                                                                            override

    Creates an action execution dialog. .

void handleCloseAppRejected (QString rejectmsg) override
    React on a rejected application close request (with some feedback message). .

void _initialize (sh::base::SingletonInitializer *singletonInitializer)
    Initializes the main window. For custom initialization logic, see MainWindow.initialize. .

void fileViewsReloadAll ()
    Reload all content in each file view.

```

**void onFileViewOptionsChanged** (std::function<void> int  
> *fcn*, QObject \**owner* = 0) Sets a handler for some view options in a file view changed (optionally bound to an owner lifetime).

**void onCurrentFileViewChanged** (std::function<void>  
> *fcn*) QObject \**owner* = 0) Sets a handler for the currently active file view changed (optionally bound to an owner lifetime).

**void onFileViewCountChanged** (std::function<void>  
> *fcn*) QObject \**owner* = 0) Sets a handler for the number of file views changed (optionally bound to an owner lifetime).

**void onCurrentDirectoryChanged** (std::function<void>  
> *fcn*) QObject \**owner* = 0) Sets a handler for the current directory in the active file view changed (optionally bound to an owner lifetime).

**void onGlobalViewOptionsChanged** (std::function<void>  
> *fcn*) QObject \**owner* = 0) Sets a handler for some global view options changed (optionally bound to an owner lifetime).

**void onCurrentProfileChanged** (std::function<void>  
> *fcn*) QObject \**owner* = 0) Sets a handler for the current profile changed (optionally bound to an owner lifetime).

**void jumpToEurl** (std::shared\_ptr<const *sh::filesystem::Eurl*> *eurl*)  
Let the current view jump to another location. .

**QString titlePattern** ()  
Returns the window title pattern.

**void setCurrentProfile** (QString *profile*)  
Sets the current profile. .

**QString currentProfile** ()  
Returns the current profile.

**void reloadProfile** ()  
Reloads the profile data.

**bool treeVisible** ()  
Returns the visibility of the directory tree.

**bool treeSticky** ()  
Returns if the directory tree follows the active file view.

**bool fileDetailsPanelVisible** ()  
Returns the visibility of the file details panel.

**std::shared\_ptr<sh::ui::ActionExecutionInfoPanel> addInfoPanel** (*sh::actions::ActionExecutionInfo* \**info*, QColor *color* = QColor())  
Creates a new action execution panel.  
Let this smart pointer die for removing it.

**std::shared\_ptr<ActionExecutionInfoPanel> showErrorPanel** ()  
Shows an error panel.

**void \_closeDirectly** ()  
Directly begin application shutdown without checks.

**bool closeApp** ()  
Check if the application may shut down now, and if so, initiate it.

bool **isCloseable** (QString \*msg = nullptr)  
 Check if the application may shut down now.

void **handleClosed** ()  
 Do some cleanup steps just when closing starts. Implementations must call base implementation!

bool **isClosing** ()  
 Returns if the application is currently closing.

bool **rootCommand** (WebServerEngineRequest \*request, QString webadapter)  
 Returns the root ('index.html') content. .

void **setEngine** (WebServerEngine \*webserver)  
 Assigns this web module to a *sh::ui::web::WebServerEngine*. .

## Public Static Functions

bool **tryCreateMainWindow** (MainWindow \*&mainWindow)  
*WebMainWindow* \*mainWindow ()

void **openUrlOnDesktop** (QUrl url)  
 Opens a url on the user desktop, typically in the browser.  
 Note: It obeys the 'openbrowser' ui parameter.

void **setMainWindow** (MainWindow \*mainWindow)  
 Sets the main window instance. .

bool **isReady** ()  
 Returns if this process ui is ready (i.e. is created or runs headless).

bool **runsHeadless** ()  
 Returns if this process runs headless, i.e. without any ui, just for a background process.

void **\_closeDirectly** (sh::base::SingletonInitializer \*singletonInitializer)

QString **uiMode** ()  
 Returns the UI mode (e.g. qt, web).

bool **executeCommand\_byQObjectReflection** (QObject \*o, WebServerEngineRequest \*request, QString command, QString justLeft-side = QString())  
 Convenience function often used by *executeCommand()*.  
 For a QObject, it searches there for a slot with the name cmd\_{command}, with `/'s replaced by '\_'s (so, for the command "foo/getBars", it searches for slotcmd\_foo\_getBars). If it finds one, it gets executed (in a worker thread). The request is considered as handled, if so.  
 If there is such a slot, but with appended \_\_M in name, it is called in main thread!  
**Return** If it handled (answered) the command.

### Private Functions

```
std::shared_ptr<sh::ui::web::WebActionManager> actionManager ()  
void setCloseable (bool closeable, QString message = QString()) override  
    Sets the closeable state and message.  
void setCurrentEurlByNode (std::shared_ptr<sh::filesystem::FilesystemNode> node)
```

### Private Members

```
bool _treeSticky = true  
bool _detailsPanelVisible = true  
QString _currentProfile  
WebServerEngine * _webserver  
std::shared_ptr<sh::filesystem::FilesystemNode> _currentDirectory = 0  
std::shared_ptr<WebActionManager> _actionManager  
std::shared_ptr<WebDetailPanelManager> _detailPanelManager  
std::shared_ptr<WebDialogManager> _dialogManager  
std::shared_ptr<WebFileViewManager> _fileViewManager  
QString _lastCloseableState
```

### Private Slots

```
void cmd_filesystem_get_icon (WebServerEngineRequest *request)  
void cmd_filesystem_list_items (WebServerEngineRequest *request)  
void cmd_log_as_text (WebServerEngineRequest *request)  
void cmd_log_log_message (WebServerEngineRequest *request)  
void cmd_mainwindow_set_current_directory (WebServerEngineRequest *request)  
void cmd_mainwindow_shallot_icon (WebServerEngineRequest *request)  
void cmd_mainwindow_show_logview (WebServerEngineRequest *request)
```

### Private Static Attributes

```
WebMainWindow * _webMainWindow = nullptr
```



## Friends

```
friend class WebFileViewManager
friend class WebActionExecutionInfoPanel
friend class WebActionExecutionInfoDialog
friend class WebDetailPanelManager
```

```
class WebManageBookmarksDialog : public QObject, public sh::ui::web::WebDialog, public sh::ui::ManageBookmarksDialog
#include <webmanagebookmarksdialog.h> Web based manage bookmarks dialog.
```

## Public Functions

```
WebManageBookmarksDialog()
```

```
QVariant dialogProperty (QString dlgPropertyKey, QVariant deflt = QVariant())
Returns a dialog property value by key.
```

```
void setDialogProperty (QString dlgPropertyKey, QVariant value)
Sets a dialog property value for a key.
```

```
QVariantHash dialogResult ()
Returns the dialog result as hash.
```

The browser side of a web dialog can ‘accept’ a dialog while closing by setting a non-empty dialog result. This mechanism allows browser side dialogs to return data to the backend. It’s up to the backend to interpret the content of this hash.

If the dialog was cancelled, or is still open, the returned hash is empty. See also wasClosed().

```
void triggerDialogEvent (QString name, QJsonObject data = QJsonObject())
Triggers a dialog event (at the browsers).
```

They in turn typically fetch some data again in order to refresh the ui.

```
void setCloseableByUser (bool v = true)
Sets if this dialog shall be directly closeable by the user by the window decoration. Set to false if the user has to react on this dialog by clicking on some controls in the dialog body.
```

This method must be called during construction.

```
bool isCloseableByUser ()
Returns if this dialog shall be directly closeable by the user by the window decoration.
```

```
QJsonObject toJson () override
Returns the json representation as QJsonObject.
```

```
qint64 dialogId ()
Returns the dialog id.
```

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

```
bool isInitiated ()
Returns if this dialog is initialized.
```

Must be called in main thread.

```
bool wasAccepted ()
Returns if this dialog was ‘accepted’ by the user (typically by clicking some ‘OK’ button).
```

Must be called in main thread.

void **waitClosed** ()

Wait until the user closed the dialog in some way.

May be called in any thread.

void **close** ()

Closes the dialog.

Must be called in main thread.

bool **wasClosed** ()

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \***manager** ()

Returns the *DialogManager* which hosts this dialog.

### Private Slots

void **cmd\_\_add\_bookmark\_\_M** (*WebServerEngineRequest* \*request)

void **cmd\_\_delete\_bookmark** (*WebServerEngineRequest* \*request)

void **cmd\_\_get** (*WebServerEngineRequest* \*request)

void **cmd\_\_move\_bookmark\_down** (*WebServerEngineRequest* \*request)

void **cmd\_\_move\_bookmark\_to\_folder** (*WebServerEngineRequest* \*request)

void **cmd\_\_move\_bookmark\_up** (*WebServerEngineRequest* \*request)

void **cmd\_\_store** (*WebServerEngineRequest* \*request)

```
class WebManageProfilesDialog : public QObject, public sh::ui::web::WebDialog, public sh::ui::ManageProfilesDialog {
    #include <webmanageprofilesdialog.h>
    Web based manage saved settings dialog.
```

### Public Functions

**WebManageProfilesDialog** ()

QVariant **dialogProperty** (QString *dlgPropertyKey*, QVariant *deflt* = QVariant())

Returns a dialog property value by key.

void **setDialogProperty** (QString *dlgPropertyKey*, QVariant *value*)

Sets a dialog property value for a key.

QVariantHash **dialogResult** ()

Returns the dialog result as hash.

The browser side of a web dialog can ‘accept’ a dialog while closing by setting a non-empty dialog result. This mechanism allows browser side dialogs to return data to the backend. It’s up to the backend to interpret the content of this hash.

If the dialog was cancelled, or is still open, the returned hash is empty. See also wasClosed().

void **triggerDialogEvent** (QString *name*, QJsonObject *data* = QJsonObject())

Triggers a dialog event (at the browsers).

They in turn typically fetch some data again in order to refresh the ui.

void **setCloseableByUser** (bool v = true)  
 Sets if this dialog shall be directly closeable by the user by the window decoration. Set to `false` if the user has to react on this dialog by clicking on some controls in the dialog body.

This method must be called during construction.

bool **isCloseableByUser** ()  
 Returns if this dialog shall be directly closeable by the user by the window decoration.

QJsonObject **toJson** () **override**  
 Returns the json representation as QJsonObject.

qint64 **dialogId** ()  
 Returns the dialog id.  
 Each instance has an id unique in the complete Shallot process lifetime.  
 Must be called in main thread.

bool **isInit** ()  
 Returns if this dialog is initialized.  
 Must be called in main thread.

bool **wasAccepted** ()  
 Returns if this dialog was 'accepted' by the user (typically by clicking some 'OK' button).  
 Must be called in main thread.

void **waitClosed** ()  
 Wait until the user closed the dialog in some way.  
 May be called in any thread.

void **close** ()  
 Closes the dialog.  
 Must be called in main thread.

bool **wasClosed** ()  
 Returns if this dialog was closed.  
 Must be called in main thread.

*DialogManager* \***manager** ()  
 Returns the *DialogManager* which hosts this dialog.

## Private Slots

void **cmd\_\_get\_\_M** (*WebServerEngineRequest* \*request)  
 void **cmd\_\_icon** (*WebServerEngineRequest* \*request)  
 void **cmd\_\_remove\_node\_\_M** (*WebServerEngineRequest* \*request)  
 void **cmd\_\_remove\_profile\_\_M** (*WebServerEngineRequest* \*request)

**class WebModule**  
*#include <webmodule.h>* Base class for modules, which provide some functionality on top of a web server engine.

Subclassed by *sh::ui::web::WebActionManager*, *sh::ui::web::WebDetailPanelManager*,  
*sh::ui::web::WebDialogManager*, *sh::ui::web::WebFileViewManager*, *sh::ui::web::WebMainWindow*,  
*sh::ui::web::WebServerEngineDispatcher*, *sh::ui::web::WebServerEngineMainModule*

## Public Functions

**WebModule** ()

**~WebModule** ()

bool **executeCommand** (*WebServerEngineRequest* \*request, QString command)

Executes *command* as requested by the browser, if responsible.

Read parameters, check if this module is responsible for answering, and then write an answer with request.

Implementations often use *executeCommand\_byQObjectReflection()* for convenience.

**Return** If it handled (answered) the command.

bool **rootCommand** (*WebServerEngineRequest* \*request, QString webadapter)

Returns the root ('index.html') content. .

void **setEngine** (*WebServerEngine* \*webserver)

Assigns this web module to a *sh::ui::web::WebServerEngine*. .

*WebServerEngine* \***webserver** ()

Returns the *sh::ui::web::WebServerEngine* this web module is assigned to (may be nullptr).

## Public Static Functions

bool **executeCommand\_byQObjectReflection** (QObject \*o, *WebServerEngineRequest* \*request, QString command, QString justLeftSide = QString())

Convenience function often used by *executeCommand()*.

For a QObject, it searches there for a slot with the name cmd\_{command}, with ``'s replaced by '\_s (so, for the command "foo/getBars", it searches for slotcmd\_foo\_getBars). If it finds one, it gets executed (in a worker thread). The request is considered as handled, if so.

If there is such a slot, but with appended \_\_M in name, it is called in main thread!

**Return** If it handled (answered) the command.

## Private Members

*WebServerEngine* \*\_**webserver** = nullptr

**class WebOpenWithDialog** : public QObject, public *sh::ui::web::WebDialog*, public *sh::ui::OpenWithDialog*  
#include <webopenwithdialog.h> Web based open with dialog.

## Public Functions

**WebOpenWithDialog** (std::shared\_ptr<*sh::filesystem::FileSystemNode*> node,  
QList<std::shared\_ptr<*sh::tools::filetypes::OpenMethod*>> openMethods)

std::shared\_ptr<*sh::tools::filetypes::OpenMethod*> **chosenMethod** () **override**

Returns the chosen open-method (program for opening the file).

bool **rememberForMimetype** () **override**

Returns if the chosen method is checked to be remembered for the same mime-type in the future.

`std::shared_ptr<const sh::filesystem::Eurl> rememberForDirectory () override`  
 If it's checked for the chosen method to be remembered for some subdirectory in the future, it returns the *Eurl* of this subdirectory, otherwise `nullptr`.

`bool rememberForFile () override`  
 Returns if the chosen method is checked to be remembered for the same file in the future.

`QVariant dialogProperty (QString dlgPropertyKey, QVariant deflt = QVariant())`  
 Returns a dialog property value by key.

`void setDialogProperty (QString dlgPropertyKey, QVariant value)`  
 Sets a dialog property value for a key.

`QVariantHash dialogResult ()`  
 Returns the dialog result as hash.

The browser side of a web dialog can 'accept' a dialog while closing by setting a non-empty dialog result. This mechanism allows browser side dialogs to return data to the backend. It's up to the backend to interpret the content of this hash.

If the dialog was cancelled, or is still open, the returned hash is empty. See also `wasClosed()`.

`void triggerDialogEvent (QString name, QJsonObject data = QJsonObject())`  
 Triggers a dialog event (at the browsers).

They in turn typically fetch some data again in order to refresh the ui.

`void setCloseableByUser (bool v = true)`  
 Sets if this dialog shall be directly closeable by the user by the window decoration. Set to `false` if the user has to react on this dialog by clicking on some controls in the dialog body.

This method must be called during construction.

`bool isCloseableByUser ()`  
 Returns if this dialog shall be directly closeable by the user by the window decoration.

`QJsonObject toJson () override`  
 Returns the json representation as `QJsonObject`.

`std::shared_ptr<sh::filesystem::FilesystemNode> node ()`  
 Returns the file node which is later to be opened.

`QList<std::shared_ptr<tools::filetypes::OpenMethod>> openMethods ()`  
 Returns the open-methods (programs for opening the file) to present for choice.

`qint64 dialogId ()`  
 Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

`bool isInitiated ()`  
 Returns if this dialog is initialized.

Must be called in main thread.

`bool wasAccepted ()`  
 Returns if this dialog was 'accepted' by the user (typically by clicking some 'OK' button).

Must be called in main thread.

`void waitClosed ()`  
 Wait until the user closed the dialog in some way.

May be called in any thread.

void **close** ()

Closes the dialog.

Must be called in main thread.

bool **wasClosed** ()

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \***manager** ()

Returns the *DialogManager* which hosts this dialog.

### Private Slots

void **cmd\_\_check\_another\_ancestor\_\_M** (*WebServerEngineRequest* \*request)

void **cmd\_\_check\_custom\_opener\_\_M** (*WebServerEngineRequest* \*request)

void **cmd\_\_open\_method\_icon\_\_M** (*WebServerEngineRequest* \*request)

void **cmd\_\_open\_methods\_\_M** (*WebServerEngineRequest* \*request)

**class WebServerEngine**

*#include <webservengine.h>* Abstract base class for a web server engine.

Such an engine implements an http server which a browser can connect to.

### Public Functions

**WebServerEngine** (bool *withMainModule*, int *minport*, int *maxport*, QString *externalUrl*,  
QString *dispatcherUrl*, QByteArray *wtoken*)

**~WebServerEngine** ()

QUrl **url** () = 0

Returns the home url of this engine (where the end user can point the browser to). .

QUrl **externalUrl** ()

Returns the external root url for creating full externally valid urls (even behind reverse proxies).

Returns *url()* if no external root url is specified.

Change this by the cmdline parameter '*externalUrl*'.

QUrl **dispatcherUrl** ()

Returns the internal root url of the dispatcher (workers only).

QByteArray **dispatcherWtoken** ()

Returns the wtoken got from the dispatcher (workers only).

QUrl **getFullCommandUrl** (QString *command*, *WebCommandData* *data*, bool *external*)

Generates a url path for a command.

Typically only needed for full url generation on server side.

void **triggerEvent** (QString *name*, QJsonObject *data*)

Triggers an event (at the clients).

void **triggerEvent** (QString *name*, QJsonValue *data* = QJsonValue::Undefined)

**QString getConfigValue** (QString *key*)  
Returns the value for a runtime configuration key.

**void setConfigValue** (QString *key*, QString *value*)  
Sets the runtime configuration key to a value (triggering a event updating the clients).

**void getStaticFile** (*WebServerEngineRequest* \**request*, QString *path*)  
Returns a static file content (for the app itself) on a request.

**QUrl rebaseUrl** (QUrl *rootUrl*, QUrl *url*)  
Returns a url rebased to another engine's root url.  
  
Used for dispatching to workers.

**void addAndPreserveModule** (std::shared\_ptr<*sh::ui::web::WebModule*> *module*)  
Plugs a *WebModule* into the engine and holds a pointer to it. Unplug it by calling *removeModule()*.

**void addModule** (std::shared\_ptr<*sh::ui::web::WebModule*> *module*)  
Plugs a *WebModule* into the engine. Unplug it by either delete *module* (drop all pointers to it) or by calling *removeModule()*.

**void removeModule** (*sh::ui::web::WebModule* \**module*)  
Unplugs a *WebModule* from the engine.

**qint64 currentTimestamp** ()  
Returns the current engine timestamp.  
  
It's not related to real time, but just a value which is higher each time you query it.  
  
This is used for coordination of some time-order sensitive operations on browser side.

## Public Static Functions

*WebServerEngine* \***createDispatcherEngine** ()  
Creates and returns a web engine for a dispatcher.

*WebServerEngine* \***createWorkerEngine** (QString *dispatcherUrl*, QByteArray *wtoken*)  
Creates and returns a web engine for a worker.

### Parameters

- *dispatcherUrl*: The root url of the dispatcher.

## Private Members

**const** QJsonObject **\_jok**

QMutex **\_configValuesMutex**

QMutex **\_modulesMutex**

int **\_minport**

int **\_maxport**

QUrl **\_externalurl**

QUrl **\_dispatcherurl**

QMap<QString, QString> **\_configValues**

QList<std::weak\_ptr<*sh::ui::web::WebModule*>> **\_modules**

QList<std::shared\_ptr<*sh::ui::web::WebModule*>> **\_smodules**

```
std::shared_ptr<sh::ui::web::WebServerEngineMainModule> _mainmodule
QString _webstaticpath
qint64 _currentTimestamp = 1
QByteArray _wtoken
```

### Private Static Functions

```
WebServerEngine *_createEngine (bool withMainModule, QString dispatcherUrl, QByteArray
                                wtoken)
```

### Friends

```
friend class WebServerEngineMainModule

class WebServerEngineDispatcher : public QObject, public sh::ui::web::WebModule
#include <webserverenginedispatcher.h> The web serve engine dispatcher module, used in dispatcher
mode for providing a portal url which starts Shallot instances on request and internally redirects to them.
```

### Public Functions

```
WebServerEngineDispatcher ()
~WebServerEngineDispatcher ()

void stop ()
    Stops the dispatcher.

void setEngine (WebServerEngine *webserver)
    Assigns this web module to a sh::ui::web::WebServerEngine. .

WebServerEngine *webserver ()
    Returns the sh::ui::web::WebServerEngine this web module is assigned to (may be nullptr).
```

### Public Static Functions

```
WebMainWindow *handleDispatching (std::function<WebMainWindow*> QString dispatcherUrl, QByteArray wtoken
    > createWndFctHandles dispatching and creates a new main window (in child processes) on demand
    via a given factory function.
```

```
void doInitialize ()
```

```
void doShutdown ()
```

```
bool executeCommand_byQObjectReflection (QObject *o, WebServerEngineRequest *re-
    quest, QString command, QString justLeft-
    side = QString())
```

Convenience function often used by *executeCommand()*.

For a QObject, it searches there for a slot with the name cmd\_{command}, with ``'s replaced by '\_s (so, for the command "foo/getBars", it searches for slotcmd\_foo\_getBars). If it finds one, it gets executed (in a worker thread). The request is considered as handled, if so.

If there is such a slot, but with appended \_\_M in name, it is called in main thread!



**Return** If it handled (answered) the command.

## Private Functions

bool **compareHashRedirectorWebworkerUrl** (QByteArray *hash*, QString *url*, QString *wtoken*)

Compares a hash for a redirector webworker url to other data.

*WorkerProcess* \***spawnNewWorker** (*WebServerEngineRequest* \**request*)

Spawns a new worker.

Does not execute any requests on it.

*WorkerProcess* \***findWorkerForWtoken** (QString *wtoken*)

Returns the worker for a wtoken.

QString **wtokenFromRequestCookie** (*WebServerEngineRequest* \**request*)

Returns the wtoken from a request (typically from cookie).

bool **executeCommand** (*WebServerEngineRequest* \**request*, QString *command*) **override**

Executes *command* as requested by the browser, if responsible.

Read parameters, check if this module is responsible for answering, and then write an answer with *request*.

Implementations often use *executeCommand\_byQObjectReflection()* for convenience.

**Return** If it handled (answered) the command.

bool **rootCommand** (*WebServerEngineRequest* \**request*, QString *webadapter*) **override**

Returns the root ('index.html') content. .

## Private Members

std::shared\_ptr<*WebServerEngine*> **\_webserver**

QMutex **\_workersmutex**

QHash<QString, *WorkerProcess*\*> **\_workers**

*JanitorThread* **\_janitor**

## Private Slots

void **cmd\_\_drop\_worker\_\_M** (*WebServerEngineRequest* \**request*)

void **cmd\_\_set\_last\_user\_activity** (*WebServerEngineRequest* \**request*)

void **cmd\_\_set\_prevent\_close\_message** (*WebServerEngineRequest* \**request*)

void **cmd\_\_working\_for** (*WebServerEngineRequest* \**request*)

### Private Static Functions

void **start** (QUrl &*url*)  
Starts the dispatcher engine, accepting http connections and internally handling child processes.

QByteArray **hashRedirectorWebworkerUrl** (QString *url*, QString *wtoken*)  
Compute a hash for a redirector webworker url.

### Private Static Attributes

std::shared\_ptr<*WebServerEngineDispatcher*> **\_dispatcher** = nullptr

QString **\_dispatcherId** = QString::fromLatin1(*sh::tools::Misc::generateUniqueHash*())

QString **\_dispatcherCookieName** = QUrl::toPercentEncoding(QString("shallot\_%1").arg(*\_dispatcherId*)).toLocal8Bit()

**class JanitorThread** : public QThread

### Public Functions

**JanitorThread** (*WebServerEngineDispatcher* \**dispatcher*)

void **run** () **override**

### Private Functions

qint64 **machineMemory** ()  
Returns the amount of memory this machine provides (in bytes).  
Returns -1 if this is not supported on the target platform or it failed.

void **stopWorker** (*WorkerProcess* \**worker*, QString *reason*)

int **detectGoodMaxNumberWorkersByMachineCapabilities** ()

QList<*WorkerProcess*\*> **stopWorkers** (QList<*WorkerProcess*\*> *workers*, int *stopcount*,  
QString *reason*)

### Private Members

*WebServerEngineDispatcher* \***\_dispatcher**

double **\_allMemory**

QHash<*WorkerProcess*\*, double> **\_workerMemoryUsagesLastSeconds**

**class WorkerProcess** : public QProcess  
A internal Shallot worker process driving one Shallot session.

## Public Functions

QByteArray **wtoken** ()

The wtoken (used for association and security).

QUrl **rootUrl** ()

The worker root url.

void **request** (*WebServerEngineRequest \*request*)

Make a request to the worker.

qint64 **workerPid** ()

Returns the process id of the worker.

qint64 **memoryUsage** ()

Returns the current memory usage of this worker (in bytes).

Returns -1 if this is not supported on the target platform or it failed.

QDateTime **lastBrowserHeartbeat** ()

Returns when a browser was seen connected to this worker last time.

QDateTime **lastUserInteraction** ()

Returns when a user has interacted with this worker last time.

QString **clientHost** ()

Returns the client host this worker is serving.

It's not guaranteed to have some specific content (maybe an ip address), but is equal for the same host, some one can count how many hosts work for one particular client host.

QString **preventCloseMessage** ()

Returns a reason message why this worker currently should not be closed (or "" if closing is fine).

See also `isClosingInappropriate()`.

Note: This is reported asynchronously. You must not rely on its precise information, but solely use it for monitoring.

bool **isClosingFine** ()

Returns if it is fine to stop this worker (or if there is a good reason to not do).

See also `preventCloseMessage()`.

Note: This is reported asynchronously. You must not rely on its precise information, but solely use it for monitoring.

## Private Functions

**WorkerProcess** (*WebServerEngineDispatcher \*dispatcher*, QByteArray *wtoken*, QString *clientHost*, QStringList *acceptedLanguages*)

bool **waitOnline** ()

Waits until the worker is online and ready for requests (or dead).

void **initialize** (QString *rooturl*, qint64 *pid*)

Initializes the worker instance (which exists on the dispatcher side!) by setting some final data.

void **gotBrowserHeartbeat** ()

Refreshes the browser heartbeat timestamp.

See `lastBrowserHeartbeat()`.

void **gotUserInteraction** (int *value*)  
Refreshes the user interaction timestamp.  
See [\*lastUserInteraction\(\)\*](#).

void **setPreventCloseMessage** (QString *message*)  
Sets the preventCloseMessage (empty: closeable).

## Private Members

*WebServerEngineDispatcher* \*\_dispatcher  
QByteArray \_wtoken  
QUrl \_rooturl  
QMutex \_mutex  
QWaitCondition \_cnd\_initd  
qint64 \_pid = -1  
QDateTime \_lastBrowserHeartbeat  
QDateTime \_lastUserInteraction  
QString \_clientHost  
QString \_preventCloseMessage  
QRegularExpression \_reProcStatus

## Friends

**friend class** WebServerEngineDispatcher  
**class** WebServerEngineMainModule : public QObject, public [\*sh::ui::web::WebModule\*](#)  
*#include <webserverengine.h>* The web serve engine main module, providing some base services like events.

## Public Functions

**WebServerEngineMainModule** ()  
**~WebServerEngineMainModule** ()  
bool **executeCommand** (*WebServerEngineRequest* \*request, QString command) **override**  
Executes command as requested by the browser, if responsible.  
Read parameters, check if this module is responsible for answering, and then write an answer with request.  
Implementations often use [\*executeCommand\\_byQObjectReflection\(\)\*](#) for convenience.  
**Return** If it handled (answered) the command.  
bool **rootCommand** (*WebServerEngineRequest* \*request, QString webadapter) **override**  
Returns the root ('index.html') content. .  
void **triggerEvent** (QString name, QString data)  
Triggers an event on browser side.

void **setEngine** (*WebServerEngine* \*webserver)

Assigns this web module to a *sh::ui::web::WebServerEngine*. .

*WebServerEngine* \*webserver ()

Returns the *sh::ui::web::WebServerEngine* this web module is assigned to (may be nullptr).

## Public Static Functions

bool **executeCommand\_byQObjectReflection** (QObject \*o, *WebServerEngineRequest* \*request, QString command, QString justLeftSide = QString())

Convenience function often used by *executeCommand()*.

For a QObject, it searches there for a slot with the name cmd\_{command}, with ``'s replaced by '\_s (so, for the command "foo/getBars", it searches for slotcmd\_foo\_getBars). If it finds one, it gets executed (in a worker thread). The request is considered as handled, if so.

If there is such a slot, but with appended \_\_M in name, it is called in main thread!

**Return** If it handled (answered) the command.

## Private Functions

qint64 **lastRequestTime** ()

## Private Members

qint64 **\_lastRequestTime**

QMutex **\_mutex\_lastRequestTime**

QList<*EventEntry*\*> **\_eventEntries**

quint64 **\_eventEntriesLastId** = 0

QMutex **\_mutex\_eventEntries**

QWaitCondition **\_eventEntriesCondition**

*EventEntryCleanupThread* **\_eventEntryCleanupThread**

*StopWhenIdleThread* **\_stopWhenIdleThread**

## Private Slots

void **cmd\_dispatcher\_toclient\_heartbeat** (*WebServerEngineRequest* \*request)

void **cmd\_events\_get\_next** (*WebServerEngineRequest* \*request)

### Private Static Attributes

QRegularExpression **\_restatic**

**class EventEntry**

Data for one occurred event.

### Public Functions

**EventEntry** (qint64 *id*, qint64 *time*, QString *name*, QString *data*)

### Public Members

**const** qint64 **id**

**const** qint64 **time**

**const** QString **name**

**const** QString **data**

### Friends

**friend class** EventEntryCleanupThread

**class EventEntryCleanupThread**: **public** QThread

Thread for cleaning up old *EventEntry* instances.

### Public Functions

**EventEntryCleanupThread** (*WebServerEngineMainModule* \**module*)

### Private Members

*WebServerEngineMainModule* \***module**

**class StopWhenIdleThread**: **public** QThread

Thread for stopping the application if the remote side is gone.

### Public Functions

**StopWhenIdleThread** (*WebServerEngineMainModule* \**module*)

## Private Members

*WebServerEngineMainModule* \*module

**class WebServerEngineRequest**

#include <webserverengine.h> A web request in a *WebServerEngine*.

Override this together with *WebServerEngine*.

## Public Functions

**WebServerEngineRequest** (QString *url*, QString *clientHost*)

QUrl **url** ()

Returns the requested url.

Note: The url should be considered as opaque, since the exact structure depends on the engine implementation.

QString **clientHost** ()

Returns the client host address.

QString **path** ()

Returns the path part of the requested url.

Note: The url should be considered as opaque, since the exact structure depends on the engine implementation.

QString **data** (QString *key*)

Returns data passed as parameter along with the request by *key*.

int **responseCode** ()

Returns the answered (http) response code.

QString **responseMimeType** ()

Returns the mimetype of the answer content.

QByteArray **response** ()

Returns the answer content.

bool **responseSet** ()

Returns if a response was set.

void **setResponse** (QByteArray *response*, QString *mimetype*, int *responseCode* = *HTTP\_OK*)

Sets the response.

void **setResponse** (QJsonArray *response*, int *responseCode* = *HTTP\_OK*)

void **setResponse** (QJsonObject *response*, int *responseCode* = *HTTP\_OK*)

QString **headerData** (QString *key*) = 0

Returns an http header value by *key*. .

QString **getCookie** (QString *key*) = 0

Returns an http cookie stored on browser side.

Note: This only works on dispatcher level and not in usual *WebModule* commands.

void **setCookie** (QString *key*, QString *value*, int *maxAgeSecs* = -1) = 0

Sets an http cookie to be returned to the browser side.

Note: This only works on dispatcher level and not in usual *WebModule* commands.

### Public Static Attributes

```
const int HTTP_OK = 200
const int HTTP_SEE_OTHER = 303
const int HTTP_NOT_FOUND = 404
const int HTTP_INTERNAL_SERVER_ERROR = 500
const int HTTP_BAD_GATEWAY = 502
```

### Private Members

```
QUrl _url
QString _clientHost
QString _path
QMap<QString, QString> _getdata
int _responseCode = HTTP_NOT_FOUND
QString _responseMimeType = "application/octet-stream"
QByteArray _response
bool _responseSet = false
```

```
class WebStoreProfileDialog : public QObject, public sh::ui::web::WebDialog, public sh::ui::StoreProfileDialog
#include <webstoreprofiledialog.h> Web based save settings dialog.
```

### Public Functions

```
WebStoreProfileDialog (std::shared_ptr<const sh::filesystem::Eurl> eurl)

QList<SettingEntry> checkedSettings () override
    Returns the list of settings the user has checked to store.

QString profileName () override
    Returns the profile the user has chosen to store settings for.

bool withSubDirectories () override
    Returns if the user has chosen to apply the checked settings also for subdirectories.

QStringList inheritsFrom () override
    Returns the list of profiles the user has chosen to inherit from.

bool hasGlobal () override
    Returns if the user has chosen to apply the checked settings for each directory (instead of the current directory).

QVariant dialogProperty (QString dlgPropertyKey, QVariant deflt = QVariant())
    Returns a dialog property value by key.

void setDialogProperty (QString dlgPropertyKey, QVariant value)
    Sets a dialog property value for a key.

QVariantHash dialogResult ()
    Returns the dialog result as hash.
```



The browser side of a web dialog can ‘accept’ a dialog while closing by setting a non-empty dialog result. This mechanism allows browser side dialogs to return data to the backend. It’s up to the backend to interpret the content of this hash.

If the dialog was cancelled, or is still open, the returned hash is empty. See also `wasClosed()`.

void **triggerDialogEvent** (QString *name*, QJsonObject *data* = QJsonObject())

Triggers a dialog event (at the browsers).

They in turn typically fetch some data again in order to refresh the ui.

void **setCloseableByUser** (bool *v* = true)

Sets if this dialog shall be directly closeable by the user by the window decoration. Set to `false` if the user has to react on this dialog by clicking on some controls in the dialog body.

This method must be called during construction.

bool **isCloseableByUser** ()

Returns if this dialog shall be directly closeable by the user by the window decoration.

QJsonObject **toJson** () **override**

Returns the json representation as QJsonObject.

std::shared\_ptr<const *sh::filesystem::Eurl*> **eurl** ()

Returns the current directory this dialog shall work on.

qint64 **dialogId** ()

Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

bool **isInitied** ()

Returns if this dialog is initialized.

Must be called in main thread.

bool **wasAccepted** ()

Returns if this dialog was ‘accepted’ by the user (typically by clicking some ‘OK’ button).

Must be called in main thread.

void **waitClosed** ()

Wait until the user closed the dialog in some way.

May be called in any thread.

void **close** ()

Closes the dialog.

Must be called in main thread.

bool **wasClosed** ()

Returns if this dialog was closed.

Must be called in main thread.

*DialogManager* \***manager** ()

Returns the *DialogManager* which hosts this dialog.

## Private Slots

```
void cmd__get__M(WebServerEngineRequest *request)
```

```
void cmd__store_mode__M(WebServerEngineRequest *request)
```

```
class WebTuningDialog : public QObject, public sh::ui::web::WebDialog, public sh::ui::TuningDialog  
#include <webtuningdialog.h> Web based tuning dialog.
```

## Public Functions

```
WebTuningDialog()
```

```
QVariant dialogProperty (QString dlgPropertyKey, QVariant deflt = QVariant())
```

Returns a dialog property value by key.

```
void setDialogProperty (QString dlgPropertyKey, QVariant value)
```

Sets a dialog property value for a key.

```
QVariantHash dialogResult ()
```

Returns the dialog result as hash.

The browser side of a web dialog can ‘accept’ a dialog while closing by setting a non-empty dialog result. This mechanism allows browser side dialogs to return data to the backend. It’s up to the backend to interpret the content of this hash.

If the dialog was cancelled, or is still open, the returned hash is empty. See also `wasClosed()`.

```
void triggerDialogEvent (QString name, QJsonObject data = QJsonObject())
```

Triggers a dialog event (at the browsers).

They in turn typically fetch some data again in order to refresh the ui.

```
void setCloseableByUser (bool v = true)
```

Sets if this dialog shall be directly closeable by the user by the window decoration. Set to `false` if the user has to react on this dialog by clicking on some controls in the dialog body.

This method must be called during construction.

```
bool isCloseableByUser ()
```

Returns if this dialog shall be directly closeable by the user by the window decoration.

```
QJsonObject toJson () override
```

Returns the json representation as QJsonObject.

```
qint64 dialogId ()
```

Returns the dialog id.

Each instance has an id unique in the complete Shallot process lifetime.

Must be called in main thread.

```
bool isInitiated ()
```

Returns if this dialog is initialized.

Must be called in main thread.

```
bool wasAccepted ()
```

Returns if this dialog was ‘accepted’ by the user (typically by clicking some ‘OK’ button).

Must be called in main thread.

```
void waitClosed()
    Wait until the user closed the dialog in some way.

    May be called in any thread.

void close()
    Closes the dialog.

    Must be called in main thread.

bool wasClosed()
    Returns if this dialog was closed.

    Must be called in main thread.

DialogManager *manager()
    Returns the DialogManager which hosts this dialog.
```

## Private Members

```
QMutex _cfgvaluesmutex
QWaitCondition _cfgvaluescondition
bool _cfgvaluesinitd = false
QList<std::shared_ptr<sh::configuration::ConfigurationValue>> _cfgvalues
```

## Private Slots

```
void cmd__change_configurationvalue (WebServerEngineRequest *request)
void cmd__get_configurationvalues (WebServerEngineRequest *request)
```

## Private Static Functions

```
QString categoryToName (sh::configuration::ConfigurationCategory category)
QString valueTypeToName (sh::configuration::ConfigurationValueType *valueType)
QJsonObject configurationValueToJsonObject (std::shared_ptr<sh::configuration::ConfigurationValue>
                                             cfgeval)
```

**namespace engines**  
Web server low-level engine implementations.

### 10.2.29 Namespace sh::ui::web::engines

**namespace engines**  
Web server low-level engine implementations.

## 10.3 Struct list

### 10.3.1 Struct `sh::actions::ActionsManager::GDAHstruct`

**struct** `sh::actions::ActionsManager::GDAHstruct`

Internal data structure used for `_getDefaultAction_helper()`;

#### Public Functions

`GDAHstruct` ( ) = default

`GDAHstruct` (const *GDAHstruct* &o) = default

#### Public Members

*sh::tools::AtomicCounter* **counter**

std::function<void (std::shared\_ptr<*sh::actions::ActionActionItem*>) > **callback**

int **currentValue** = 0

QList<std::shared\_ptr<*sh::actions::AbstractActionItem*>> **initialActionList**

std::shared\_ptr<*sh::actions::AbstractActionItem*> **currentAction** = 0

### 10.3.2 Struct `sh::base::Logger::LogMessage`

**struct** `sh::base::Logger::LogMessage`

A single log message.

#### Public Members

QString **message**

*LogSeverity* **severity**

QString **source**

QDateTime **time**

### 10.3.3 Struct `sh::filesystemhandlers::GnomeIODevicesFilesystemHandler::GnomeIODevice`

**struct** `sh::filesystemhandlers::GnomeIODevicesFilesystemHandler::GnomeIODevice`

## Public Functions

**GnomeIODevice** (QString *name*, QString *displayname*, QString *url*, bool *isMounted*)

## Public Members

QString **name**

QString **displayname**

QString **url**

bool **isMounted**

### 10.3.4 Struct `sh::filesystemhandlers::GnomeIOFilesystemHandler::MountOperationsCallbackData`

```
struct sh::filesystemhandlers::GnomeIOFilesystemHandler::MountOperationsCallbackData
```

## Public Functions

**MountOperationsCallbackData** (*sh::actions::ActionExecutionInfo* \**info*, size\_t *index*, void \**mountoperation*, QString *address*)

## Public Members

*sh::actions::ActionExecutionInfo* \***info**

size\_t **index**

void \***mountoperation**

QString **address**

### 10.3.5 Struct `sh::scripting::pythoninterop::CallNativeHelperMethodWithPyObjectArguments`

```
template<typename T, T>
struct CallNativeHelperMethodWithPyObjectArguments
```

### 10.3.6 Struct `sh::scripting::pythoninterop::CallNativeHelperMethodWithPyObjectArguments<R(*) (T *, Args...), MF>`

```
template<typename T, typename R, typename ... Args, R(*) (T *, Args...) MF> sh::scripting::pythoninterop::CallNativeHelperMethodWithPyObjectArguments<R(*) (T *, Args...), MF>
```

### Public Static Functions

PyObject \***call** (PyObject \**a*, PyObject \**b*)

#### 10.3.7 Struct sh::scripting::pythoninterop::CallNativeHelperMethodWithPyObjectArguments<void(\*) (T \*, Args...), MF >

```
template<typename T, typename ... Args, void(*) (T *, Args...) MF> sh::scripting::pythonint
```

### Public Static Functions

PyObject \***call** (PyObject \**a*, PyObject \**b*)

#### 10.3.8 Struct sh::scripting::pythoninterop::CallNativeMethodWithPyObjectArguments

```
template<typename T, T>  
struct CallNativeMethodWithPyObjectArguments
```

#### 10.3.9 Struct sh::scripting::pythoninterop::CallNativeMethodWithPyObjectArguments<R(T::\*)(Args...), MF >

```
template<typename T, typename R, typename ... Args, R(T::*)(Args...) MF> sh::scripting::pyt
```

### Public Static Functions

PyObject \***call** (PyObject \**a*, PyObject \**b*)

#### 10.3.10 Struct sh::scripting::pythoninterop::CallNativeMethodWithPyObjectArguments<R(T::\*)(Args...) const, MF >

```
template<typename T, typename R, typename ... Args, R(T::*)(Args...) const MF> sh::scripti
```

### Public Static Functions

PyObject \***call** (PyObject \**a*, PyObject \**b*)

#### 10.3.11 Struct sh::scripting::pythoninterop::CallNativeMethodWithPyObjectArguments<void(T::\*)(Args...) const, MF >

```
template<typename T, typename ... Args, void(T::*)(Args...) const MF> sh::scripting::pytho
```

**Public Static Functions**

PyObject \***call** (PyObject \**a*, PyObject \**b*)

### 10.3.12 Struct `sh::scripting::pythoninterop::CallNativeMethodWithPyObjectArguments<void(T::*)(Args...), MF >`

```
template<typename T, typename ... Args, void(T::*)(Args...) MF> sh::scripting::pythoninterop::
```

**Public Static Functions**

PyObject \***call** (PyObject \**a*, PyObject \**b*)

### 10.3.13 Struct `sh::scripting::pythoninterop::FunctionBindTuple`

```
template<int mode, uint N>
struct FunctionBindTuple
```

### 10.3.14 Struct `sh::scripting::pythoninterop::FunctionBindTuple< 0, 0 >`

```
template<>
struct sh::scripting::pythoninterop::FunctionBindTuple<0,0>
```

**Public Static Functions**

```
template<typename R, typename T, typename ...ArgsF, typename ...ArgsT, typename ...Args>
R applyTuple (T *pObj, R (T::*f)) ArgsF...
, const std::tuple<ArgsT...>&, Args... args
```

### 10.3.15 Struct `sh::scripting::pythoninterop::FunctionBindTuple< 0, N >`

```
template<uint N>
struct sh::scripting::pythoninterop::FunctionBindTuple<0,N>
```

**Public Static Functions**

```
template<typename R, typename T, typename ...ArgsF, typename ...ArgsT, typename ...Args>
R applyTuple (T *pObj, R (T::*f)) ArgsF...
, const std::tuple<ArgsT...> &t, Args... args
```

### 10.3.16 Struct `sh::scripting::pythoninterop::FunctionBindTuple< 1, 0 >`

```
template<>
struct sh::scripting::pythoninterop::FunctionBindTuple<1, 0>
```

#### Public Static Functions

```
template<typename R, typename T, typename ...ArgsF, typename ...ArgsT, typename ...Args>
R applyTuple (T *pObj, R (*f)) T*, ArgsF...
, const std::tuple<ArgsT...>&, Args... args
```

### 10.3.17 Struct `sh::scripting::pythoninterop::FunctionBindTuple< 1, N >`

```
template<uint N>
struct sh::scripting::pythoninterop::FunctionBindTuple<1, N>
```

#### Public Static Functions

```
template<typename R, typename T, typename ...ArgsF, typename ...ArgsT, typename ...Args>
R applyTuple (T *pObj, R (*f)) T*, ArgsF...
, const std::tuple<ArgsT...> &t, Args... args
```

### 10.3.18 Struct `sh::scripting::pythoninterop::PyObjectTupleToNativeTuple`

```
template<int pos, typename ...Args>
struct PyObjectTupleToNativeTuple
```

### 10.3.19 Struct `sh::scripting::pythoninterop::PyObjectTupleToNativeTuple< pos, T, Args... >`

```
template<int pos, class T, class ...Args>
struct sh::scripting::pythoninterop::PyObjectTupleToNativeTuple<pos, T, Args...>
```

#### Public Static Functions

```
std::tuple<T, Args...> toNativeTuple (PyObject *a, PyObject *b)
```

### 10.3.20 Struct `sh::scripting::pythoninterop::PyObjectTupleToNativeTuple< pos >`

```
template<int pos>
struct sh::scripting::pythoninterop::PyObjectTupleToNativeTuple<pos>
```



### Public Static Functions

```
std::tuple toNativeTuple (PyObject*, PyObject*)
```

#### 10.3.21 Struct `sh::scripting::pythoninterop::ScriptedMethodProxy`

```
template<typename D, typename T, T>
struct ScriptedMethodProxy
```

#### 10.3.22 Struct `sh::scripting::pythoninterop::ScriptedMethodProxy<T, std::function<R(Args...)> T::*, MF >`

```
template<typename T, typename R, typename ...Args, std::function<R (Args...) > T::* MF>
struct sh::scripting::pythoninterop::ScriptedMethodProxy<T, std::function<R (Args...) >
                                                    T::*,
                                                    MF>
```

### Public Static Functions

```
void setImplementation (T *inst, std::function<R> Args...
    > vfct
PyObject *pyObjectFunctionSetImplementation (PyObject *a, PyObject *b)
```

#### 10.3.23 Struct `sh::scripting::shallot_ShallotObject`

```
struct sh::scripting::shallot_ShallotObject
    A wrapper around a native object for the Python world.
```

### Public Members

```
PyObject_HEAD void * nativeObject
std::shared_ptr<void> sharedNativeObject
bool isReallyShared
```

#### 10.3.24 Struct `sh::tools::filetypes::FreedesktopOrgToolsOpenMethodDeterminationStrategy::Application`

```
struct sh::tools::filetypes::FreedesktopOrgToolsOpenMethodDeterminationStrategy::Application
```

### Public Members

QString **name**  
QStringList **mimetypes**  
QString **command**  
QStringList **commandargs**  
QIcon **icon**  
bool **hidden**

## 10.3.25 Struct `sh::tools::filetypes::OpenMethod`

**struct** `sh::tools::filetypes::OpenMethod`  
Commandline and infos for opening a file with an external program.

### Public Functions

**OpenMethod** (QString *name*, QString *command*, QStringList *arguments*, QIcon *icon* = QIcon(), int *precedence* = 0)

### Public Members

const QString **name**  
const QString **command**  
const QStringList **arguments**  
const QIcon **icon**  
const int **precedence**

## 10.3.26 Struct `sh::ui::ManageBookmarksDialog::BM`

**struct** `sh::ui::ManageBookmarksDialog::BM`: **public** enable\_shared\_from\_this<BM>  
Subclassed by `sh::ui::ManageBookmarksDialog::BM_Bookmark`, `sh::ui::ManageBookmarksDialog::BM_Folder`

### Public Functions

std::shared\_ptr<BM\_Folder> **asFolder** () = 0  
std::shared\_ptr<BM\_Bookmark> **asBookmark** () = 0  
bool **isFolder** () = 0

### 10.3.27 Struct `sh::ui::ManageBookmarksDialog::BM_Bookmark`

```
struct sh::ui::ManageBookmarksDialog::BM_Bookmark : public sh::ui::ManageBookmarksDialog::BM
```

#### Public Functions

```
BM_Bookmark (std::shared_ptr<sh::tools::Bookmark> bookmark)  
std::shared_ptr<BM_Folder> asFolder () override  
std::shared_ptr<BM_Bookmark> asBookmark () override  
bool isFolder () override
```

#### Public Members

```
const std::shared_ptr<sh::tools::Bookmark> bookmark
```

### 10.3.28 Struct `sh::ui::ManageBookmarksDialog::BM_Folder`

```
struct sh::ui::ManageBookmarksDialog::BM_Folder : public sh::ui::ManageBookmarksDialog::BM
```

#### Public Functions

```
std::shared_ptr<BM_Folder> asFolder () override  
std::shared_ptr<BM_Bookmark> asBookmark () override  
bool isFolder () override
```

#### Public Members

```
QString name  
QList<std::shared_ptr<BM>> children
```

### 10.3.29 Struct `sh::ui::ManageProfilesDialog::Node`

```
struct sh::ui::ManageProfilesDialog::Node
```

#### Public Members

```
QString text  
QString icon  
bool isGlobal  
QString nodeId  
sh::settings::ProfileNode *pnode
```



## Symbols

`_ProjectInfo` (C++ class), 597

`_ProjectInfo::buildtimeutc` (C++ member), 597

`_ProjectInfo::homepage` (C++ member), 597

`_ProjectInfo::version` (C++ member), 597

## P

`PhonyNameDueToError::call` (C++ function), 1882, 1883

## S

`sh` (C++ type), 598

`sh::actions` (C++ type), 598, 1106

`sh::actions::AbstractActionFactory` (C++ class), 21, 598, 1106

`sh::actions::AbstractActionFactory::~AbstractActionFactory` (C++ function), 21, 598, 1106

`sh::actions::AbstractActionFactory::AbstractActionFactory` (C++ function), 21, 598, 1106

`sh::actions::AbstractActionFactory::actionAvailable` (C++ function), 21, 598, 1106

`sh::actions::AbstractActionFactory::category` (C++ member), 22, 599, 1106

`sh::actions::AbstractActionFactory::construct` (C++ function), 21, 598, 1106

`sh::actions::AbstractActionFactory::predicates` (C++ member), 22, 599, 1106

`sh::actions::AbstractActionItem` (C++ class), 22, 599, 1106

`sh::actions::AbstractActionItem::_defaultActionPrecedence` (C++ member), 23, 600, 1108

`sh::actions::AbstractActionItem::_enabled` (C++ member), 23, 600, 1108

`sh::actions::AbstractActionItem::_icon` (C++ member), 23, 600, 1108

`sh::actions::AbstractActionItem::_initialized` (C++ member), 23, 600, 1108

`sh::actions::AbstractActionItem::_initializing` (C++ member), 23, 600, 1108

`sh::actions::AbstractActionItem::_ischeckable` (C++ member), 23, 600, 1108

`sh::actions::AbstractActionItem::_ischecked` (C++ member), 23, 600, 1108

`sh::actions::AbstractActionItem::_parentAction` (C++ member), 23, 600, 1108

`sh::actions::AbstractActionItem::_reinitialize` (C++ member), 23, 600, 1108

`sh::actions::AbstractActionItem::_setParentAction` (C++ function), 23, 600, 1107

`sh::actions::AbstractActionItem::_text` (C++ member), 23, 600, 1108

`sh::actions::AbstractActionItem::_visible` (C++ member), 23, 600, 1108

`sh::actions::AbstractActionItem::AbstractActionItem` (C++ function), 23, 600, 1107

`sh::actions::AbstractActionItem::changed` (C++ function), 23, 600, 1108

`sh::actions::AbstractActionItem::defaultActionPrecedence` (C++ function), 22, 599, 1107

`sh::actions::AbstractActionItem::enabled` (C++ function), 22, 599, 1107

`sh::actions::AbstractActionItem::icon` (C++ function), 22, 599, 1107

`sh::actions::AbstractActionItem::initcondition` (C++ member), 24, 600, 1108

`sh::actions::AbstractActionItem::initializeAsync` (C++ function), 23, 600, 1107

`sh::actions::AbstractActionItem::initializeSync` (C++ function), 23, 600, 1107

`sh::actions::AbstractActionItem::isCheckable` (C++ function), 22, 599, 1107

`sh::actions::AbstractActionItem::isChecked` (C++ function), 22, 599, 1107

`sh::actions::AbstractActionItem::isInitialized` (C++ function), 23, 600, 1107

`sh::actions::AbstractActionItem::isInitializing` (C++ function), 23, 600, 1107

`sh::actions::AbstractActionItem::parentAction` (C++ function), 23, 599, 1107

`sh::actions::AbstractActionItem::setChecked` (C++ function), 22, 599, 1107

`sh::actions::AbstractActionItem::setEnabled` (C++ function), 22, 599, 1107

sh::actions::AbstractActionItem::setIconsh::actions::ActionActionItem::shortcuthintTrigger  
(C++ function), 22, 599, 1107 (C++ function), 25, 601, 1109

sh::actions::AbstractActionItem::setTextsh::actions::ActionActionItem::text  
(C++ function), 22, 599, 1107 (C++ function), 25, 601, 1109

sh::actions::AbstractActionItem::setVisiblesh::actions::ActionActionItem::visible  
(C++ function), 22, 599, 1107 (C++ function), 25, 602, 1110

sh::actions::AbstractActionItem::text sh::actions::ActionCategory (C++ class), 26,  
(C++ function), 22, 599, 1107 603, 1110

sh::actions::AbstractActionItem::visiblesh::actions::ActionCategory::\_categories  
(C++ function), 22, 599, 1107 (C++ member), 27, 604, 1111

sh::actions::ActionActionItem (C++ class), sh::actions::ActionCategory::\_icon (C++  
24, 600, 1108 member), 27, 603, 1111

sh::actions::ActionActionItem::\_setParentsActionsh::actions::ActionCategory::\_label  
(C++ function), 26, 602, 1110 (C++ member), 27, 603, 1111

sh::actions::ActionActionItem::ActionActionItemsh::actions::ActionCategory::\_name (C++  
(C++ function), 25, 601, 1109 member), 27, 603, 1111

sh::actions::ActionActionItem::changed sh::actions::ActionCategory::ActionCategory  
(C++ function), 26, 603, 1110 (C++ function), 26, 603, 1110

sh::actions::ActionActionItem::defaultActionPrecedenceActionCategory::add (C++  
(C++ function), 25, 602, 1109 function), 26, 603, 1111

sh::actions::ActionActionItem::enabled sh::actions::ActionCategory::categories  
(C++ function), 25, 602, 1109 (C++ function), 26, 603, 1111

sh::actions::ActionActionItem::execute sh::actions::ActionCategory::doInitialize  
(C++ function), 25, 601, 1109 (C++ function), 26, 603, 1111

sh::actions::ActionActionItem::icon sh::actions::ActionCategory::doShutdown  
(C++ function), 25, 602, 1109 (C++ function), 26, 603, 1111

sh::actions::ActionActionItem::initializeAsyncsh::actions::ActionCategory::getByName  
(C++ function), 26, 602, 1110 (C++ function), 26, 603, 1111

sh::actions::ActionActionItem::initializeSyncsh::actions::ActionCategory::icon (C++  
(C++ function), 26, 602, 1110 function), 26, 603, 1110

sh::actions::ActionActionItem::isCheckedsh::actions::ActionCategory::label (C++  
(C++ function), 25, 602, 1109 function), 26, 603, 1110

sh::actions::ActionActionItem::isCheckedsh::actions::ActionCategory::name (C++  
(C++ function), 25, 602, 1109 function), 26, 603, 1110

sh::actions::ActionActionItem::isInitialsh::actions::ActionDefaultPrecedenceValues  
(C++ function), 26, 602, 1110 (C++ class), 27, 604, 1111

sh::actions::ActionActionItem::isInitialsh::actions::ActionDefaultPrecedenceValues::PRECED  
(C++ function), 26, 602, 1110 (C++ member), 27, 604, 1111

sh::actions::ActionActionItem::parentActionsh::actions::ActionDefaultPrecedenceValues::PRECED  
(C++ function), 25, 602, 1110 (C++ member), 27, 604, 1111

sh::actions::ActionActionItem::setCheckedsh::actions::ActionExecutionInfo (C++  
(C++ function), 25, 602, 1110 class), 27, 604, 1111

sh::actions::ActionActionItem::setEnabledsh::actions::ActionExecutionInfo::\_checkVisibility  
(C++ function), 25, 602, 1110 (C++ function), 30, 607, 1114

sh::actions::ActionActionItem::setIcon sh::actions::ActionExecutionInfo::\_fromobj  
(C++ function), 25, 602, 1110 (C++ member), 29, 606, 1113

sh::actions::ActionActionItem::setShortcuthintsh::actions::ActionExecutionInfo::\_fromverb  
(C++ function), 25, 601, 1109 (C++ member), 29, 606, 1113

sh::actions::ActionActionItem::setText sh::actions::ActionExecutionInfo::\_head  
(C++ function), 25, 602, 1109 (C++ member), 29, 606, 1113

sh::actions::ActionActionItem::setVisiblesh::actions::ActionExecutionInfo::\_iscancelled  
(C++ function), 25, 602, 1110 (C++ member), 30, 606, 1114

sh::actions::ActionActionItem::shortcuthintsh::actions::ActionExecutionInfo::\_manualIntervent  
(C++ function), 25, 601, 1109 (C++ member), 30, 606, 1114

```

sh::actions::ActionExecutionInfo::_operation
    (C++ member), 30, 606, 1114
sh::actions::ActionExecutionInfo::_progressAll
    (C++ member), 29, 606, 1114
sh::actions::ActionExecutionInfo::_progressDone
    (C++ member), 29, 606, 1113
sh::actions::ActionExecutionInfo::_progressText
    (C++ member), 30, 606, 1114
sh::actions::ActionExecutionInfo::_toobjsh
    (C++ member), 29, 606, 1113
sh::actions::ActionExecutionInfo::_toverbsh
    (C++ member), 29, 606, 1113
sh::actions::ActionExecutionInfo::_visualProcessFeedback
    (C++ member), 29, 606, 1114
sh::actions::ActionExecutionInfo::_wasAlwaysVisible
    (C++ member), 30, 606, 1114
sh::actions::ActionExecutionInfo::~ActionExecutionInfo
    (C++ function), 27, 604, 1111
sh::actions::ActionExecutionInfo::ActionExecutionInfo
    (C++ function), 27, 604, 1111
sh::actions::ActionExecutionInfo::addChangeListener
    (C++ function), 29, 605, 1113
sh::actions::ActionExecutionInfo::cancelsh
    (C++ function), 28, 605, 1112
sh::actions::ActionExecutionInfo::changesh
    (C++ member), 30, 606, 1114
sh::actions::ActionExecutionInfo::createDialog
    (C++ function), 29, 606, 1113
sh::actions::ActionExecutionInfo::createPanel
    (C++ function), 29, 606, 1113
sh::actions::ActionExecutionInfo::detailsChanged
    (C++ function), 29, 606, 1113
sh::actions::ActionExecutionInfo::finishExecution
    (C++ function), 28, 604, 1112
sh::actions::ActionExecutionInfo::fromObjectName
    (C++ function), 28, 604, 1112
sh::actions::ActionExecutionInfo::fromVerbsh
    (C++ function), 28, 604, 1112
sh::actions::ActionExecutionInfo::head
    (C++ function), 28, 605, 1112
sh::actions::ActionExecutionInfo::headChanged
    (C++ function), 29, 606, 1113
sh::actions::ActionExecutionInfo::infoDialog
    (C++ member), 30, 606, 1114
sh::actions::ActionExecutionInfo::infoPanel
    (C++ member), 30, 606, 1114
sh::actions::ActionExecutionInfo::isCancelled
    (C++ function), 28, 605, 1112
sh::actions::ActionExecutionInfo::isManualInterventionNeeded
    (C++ function), 27, 604, 1112
sh::actions::ActionExecutionInfo::isProgressDone
    (C++ function), 28, 605, 1112
sh::actions::ActionExecutionInfo::isVisualProcessFeedback
    (C++ function), 27, 604, 1111
sh::actions::ActionExecutionInfo::manualIntervention
    (C++ function), 29, 606, 1113
sh::actions::ActionExecutionInfo::mutex
    (C++ member), 29, 606, 1113
sh::actions::ActionExecutionInfo::operation
    (C++ function), 29, 605, 1113
sh::actions::ActionExecutionInfo::progressAll
    (C++ function), 28, 605, 1112
sh::actions::ActionExecutionInfo::progressChanged
    (C++ function), 29, 606, 1113
sh::actions::ActionExecutionInfo::progressDone
    (C++ function), 28, 605, 1112
sh::actions::ActionExecutionInfo::progressText
    (C++ function), 28, 605, 1112
sh::actions::ActionExecutionInfo::respectCancel
    (C++ function), 28, 605, 1113
sh::actions::ActionExecutionInfo::setDetails
    (C++ function), 28, 604, 1112
sh::actions::ActionExecutionInfo::setHead
    (C++ function), 28, 605, 1112
sh::actions::ActionExecutionInfo::setManualIntervention
    (C++ function), 27, 604, 1112
sh::actions::ActionExecutionInfo::setProgress
    (C++ function), 28, 605, 1112
sh::actions::ActionExecutionInfo::setProgressIndeterminate
    (C++ function), 28, 605, 1112
sh::actions::ActionExecutionInfo::setVisualProcessFeedback
    (C++ function), 27, 604, 1111
sh::actions::ActionExecutionInfo::shutdown
    (C++ function), 29, 605, 1113
sh::actions::ActionExecutionInfo::stack
    (C++ member), 30, 606, 1114
sh::actions::ActionExecutionInfo::startExecution
    (C++ function), 28, 604, 1112
sh::actions::ActionExecutionInfo::toObjectName
    (C++ function), 28, 605, 1112
sh::actions::ActionExecutionInfo::toVerb
    (C++ function), 28, 604, 1112
sh::actions::ActionExecutionInfo::userfeedback
    (C++ function), 28, 605, 1112
sh::actions::ActionExecutionInfo::visualProcessFeedback
    (C++ function), 29, 606, 1113
sh::actions::ActionExecutionInfo::wasCancelled
    (C++ function), 29, 606, 1113
sh::actions::ActionExecutionInfo::withExecuteGuards
    (C++ function), 28, 605, 1113
sh::actions::ActionExecutionUserFeedback
    (C++ class), 30, 607, 1114
sh::actions::ActionExecutionUserFeedback::~ActionExecutionUserFeedback
    (C++ function), 31, 608, 1115
sh::actions::ActionExecutionUserFeedback::Choice
    (C++ class), 31, 32, 608, 1115

```

sh::actions::ActionExecutionUserFeedbackshChactéonShoActionExecutionUserFeedback::MessageB  
 (C++ function), 31, 33, 608, 1115 (C++ enumerator), 30, 607, 1114  
 sh::actions::ActionExecutionUserFeedbackshChactéonshabActionExecutionUserFeedback::multiline  
 (C++ member), 31, 33, 608, 1115 (C++ function), 31, 607, 1115  
 sh::actions::ActionExecutionUserFeedbackshChactéonsexActionExecutionUserFeedback::simpleCh  
 (C++ member), 31, 33, 608, 1115 (C++ function), 31, 607, 1115  
 sh::actions::ActionExecutionUserFeedbackshChactéons::ActionExecutionUserFeedback::simpleInp  
 (C++ class), 31, 33, 608, 1115 (C++ function), 31, 608, 1115  
 sh::actions::ActionExecutionUserFeedbackshChactéonsliActionExecutionUserFeedback::simpleMe  
 (C++ member), 32, 33, 608, 1116 (C++ function), 31, 608, 1115  
 sh::actions::ActionExecutionUserFeedbackshChactéonssseActiedExecutionUserFeedback::unixPerm  
 (C++ member), 32, 33, 608, 1116 (C++ function), 31, 607, 1115  
 sh::actions::ActionExecutionUserFeedbackshcredentialsAclalogFactory (C++ class), 34,  
 (C++ function), 31, 607, 1115 609, 1116  
 sh::actions::ActionExecutionUserFeedbackshGrádformEntActéonFactory::actionAvailable  
 (C++ class), 32, 33, 608, 1116 (C++ function), 34, 609, 1117  
 sh::actions::ActionExecutionUserFeedbackshGrádformEntActéonFGridformEntéonFactory  
 (C++ function), 32, 33, 608, 1116 (C++ function), 34, 609, 1117  
 sh::actions::ActionExecutionUserFeedbackshGrádformEntActéonEactéonEntéonset  
 (C++ function), 32, 33, 608, 1116 (C++ function), 34, 609, 1117  
 sh::actions::ActionExecutionUserFeedbackshGrádformEntActéonInstantiation (C++  
 (C++ member), 32, 33, 609, 1116 class), 34, 609, 1117  
 sh::actions::ActionExecutionUserFeedbackshGrádformEntActéonNewEntéon::actionfactory  
 (C++ function), 32, 33, 608, 1116 (C++ member), 35, 610, 1117  
 sh::actions::ActionExecutionUserFeedbackshGrádformEntActéonInstantiation::ActionInstantiat  
 (C++ class), 32, 33, 609, 1116 (C++ function), 34, 609, 1117  
 sh::actions::ActionExecutionUserFeedbackshGrádformEntActéonGridformEntéon::category  
 (C++ function), 32, 34, 609, 1116 (C++ member), 35, 610, 1117  
 sh::actions::ActionExecutionUserFeedbackshGrádformEntActéonGridformEntéon::createdAction  
 (C++ function), 32, 34, 609, 1116 (C++ member), 35, 610, 1117  
 sh::actions::ActionExecutionUserFeedbackshGrádformEntActéonInstantiation::currentDirectory  
 (C++ member), 32, 34, 609, 1116 (C++ member), 35, 610, 1117  
 sh::actions::ActionExecutionUserFeedbackshGrádformEntActéonGridformEntéon::isLookupOnCurrent  
 (C++ function), 32, 34, 609, 1116 (C++ member), 35, 610, 1117  
 sh::actions::ActionExecutionUserFeedbackshGrádformEntActéonInstantiation::positionIndex  
 (C++ member), 32, 34, 609, 1116 (C++ member), 35, 610, 1117  
 sh::actions::ActionExecutionUserFeedbackshinputBoxes::ActionInstantiation::resolveLinks  
 (C++ function), 31, 607, 1115 (C++ member), 35, 610, 1117  
 sh::actions::ActionExecutionUserFeedbackshmessageBox:ActionInstantiation::selectedNodes  
 (C++ function), 31, 607, 1115 (C++ member), 35, 610, 1117  
 sh::actions::ActionExecutionUserFeedbackshMessageBoxBAttéonInstantiation::shortcut  
 (C++ enum), 30, 607, 1114 (C++ member), 35, 610, 1117  
 sh::actions::ActionExecutionUserFeedbackshMessageBoxBAttéonsManager (C++ class), 35,  
 (C++ enumerator), 30, 607, 1114 610, 1118  
 sh::actions::ActionExecutionUserFeedbackshMessageBoxBAttéonsManager:\_actionExecutionStarte  
 (C++ enumerator), 30, 607, 1114 (C++ function), 37, 612, 1119  
 sh::actions::ActionExecutionUserFeedbackshMessageBoxBAttéonsManager::\_actionfactories  
 (C++ enumerator), 30, 607, 1114 (C++ member), 37, 612, 1120  
 sh::actions::ActionExecutionUserFeedbackshMessageBoxBAttéonsManager::\_emit\_runningActionsCh  
 (C++ enumerator), 30, 607, 1114 (C++ function), 37, 612, 1119  
 sh::actions::ActionExecutionUserFeedbackshMessageBoxBAttéonsManager::\_getActions\_raw  
 (C++ enumerator), 30, 607, 1114 (C++ function), 37, 612, 1119  
 sh::actions::ActionExecutionUserFeedbackshMessageBoxBAttéonsManager::\_getDefaultAction\_help  
 (C++ enumerator), 30, 607, 1114 (C++ function), 37, 612, 1119







(C++ function), 53, 626, 1133, 1234  
 sh::actions::common::ActionAddToBookmarks:isActionSizeCommon::ActionBookmark::isChecked  
 (C++ function), 53, 626, 1133, 1234 (C++ function), 54, 627, 1134, 1235  
 sh::actions::common::ActionAddToBookmarks:isCheckableCommon::ActionBookmark::isInitialized  
 (C++ function), 52, 625, 1132, 1234 (C++ function), 54, 627, 1134, 1236  
 sh::actions::common::ActionAddToBookmarks:isChecked:common::ActionBookmark::isInitializing  
 (C++ function), 52, 625, 1132, 1234 (C++ function), 54, 627, 1134, 1236  
 sh::actions::common::ActionAddToBookmarks:isMinimalizedCommon::ActionBookmark::parentAction  
 (C++ function), 53, 626, 1133, 1234 (C++ function), 54, 627, 1134, 1236  
 sh::actions::common::ActionAddToBookmarks:isMinimalizedCommon::ActionBookmark::setChecked  
 (C++ function), 53, 626, 1133, 1234 (C++ function), 54, 627, 1134, 1236  
 sh::actions::common::ActionAddToBookmarks:parentActionCommon::ActionBookmark::setEnabled  
 (C++ function), 53, 626, 1133, 1234 (C++ function), 54, 627, 1134, 1236  
 sh::actions::common::ActionAddToBookmarks:setCheckedCommon::ActionBookmark::setIcon  
 (C++ function), 52, 625, 1133, 1234 (C++ function), 54, 627, 1134, 1235  
 sh::actions::common::ActionAddToBookmarks:setEnabledCommon::ActionBookmark::setShortcutHint  
 (C++ function), 52, 625, 1133, 1234 (C++ function), 53, 626, 1134, 1235  
 sh::actions::common::ActionAddToBookmarks:setIcons:common::ActionBookmark::setText  
 (C++ function), 52, 625, 1132, 1234 (C++ function), 54, 627, 1134, 1235  
 sh::actions::common::ActionAddToBookmarks:setShortcutHintCommon::ActionBookmark::setVisible  
 (C++ function), 52, 625, 1132, 1233 (C++ function), 54, 627, 1134, 1236  
 sh::actions::common::ActionAddToBookmarks:setTexts:common::ActionBookmark::shortcutHint  
 (C++ function), 52, 625, 1132, 1234 (C++ function), 53, 626, 1133, 1235  
 sh::actions::common::ActionAddToBookmarks:setVisibleCommon::ActionBookmark::shortcutHintText  
 (C++ function), 52, 626, 1133, 1234 (C++ function), 53, 626, 1133, 1235  
 sh::actions::common::ActionAddToBookmarks:shortIconHintCommon::ActionBookmark::text  
 (C++ function), 52, 625, 1132, 1233 (C++ function), 53, 627, 1134, 1235  
 sh::actions::common::ActionAddToBookmarks:shortIconHintOfMongooseOnCnBookmarksVisible  
 (C++ function), 52, 625, 1132, 1233 (C++ function), 54, 627, 1134, 1236  
 sh::actions::common::ActionAddToBookmarks:texts:common::ActionBookmarkFolder  
 (C++ function), 52, 625, 1132, 1234 (C++ class), 55, 628, 1135, 1236  
 sh::actions::common::ActionAddToBookmarks:visible:common::ActionBookmarkFolder::\_setParent  
 (C++ function), 52, 626, 1133, 1234 (C++ function), 56, 629, 1136, 1237  
 sh::actions::common::ActionBookmark sh::actions::common::ActionBookmarkFolder::ActionBo  
 (C++ class), 53, 626, 1133, 1235 (C++ function), 55, 628, 1135, 1236  
 sh::actions::common::ActionBookmark::\_setParentActionCommon::ActionBookmarkFolder::addSubit  
 (C++ function), 54, 627, 1134, 1236 (C++ function), 55, 628, 1135, 1236  
 sh::actions::common::ActionBookmark::ActionBookmark:common::ActionBookmarkFolder::changed  
 (C++ function), 53, 626, 1133, 1235 (C++ function), 56, 629, 1136, 1238  
 sh::actions::common::ActionBookmark::changedActions:common::ActionBookmarkFolder::clearSub  
 (C++ function), 55, 628, 1135, 1236 (C++ function), 55, 628, 1135, 1236  
 sh::actions::common::ActionBookmark::defaultActionRecommendone:ActionBookmarkFolder::defaultT  
 (C++ function), 54, 627, 1134, 1235 (C++ function), 55, 628, 1135, 1237  
 sh::actions::common::ActionBookmark::enabledActions:common::ActionBookmarkFolder::enabled  
 (C++ function), 54, 627, 1134, 1235 (C++ function), 55, 628, 1135, 1236  
 sh::actions::common::ActionBookmark::executeActions:common::ActionBookmarkFolder::icon  
 (C++ function), 53, 626, 1133, 1235 (C++ function), 55, 628, 1135, 1236  
 sh::actions::common::ActionBookmark::iconsh::actions::common::ActionBookmarkFolder::initial  
 (C++ function), 54, 627, 1134, 1235 (C++ function), 56, 629, 1136, 1237  
 sh::actions::common::ActionBookmark::initializeAsync:common::ActionBookmarkFolder::initial  
 (C++ function), 54, 627, 1134, 1236 (C++ function), 56, 629, 1136, 1237  
 sh::actions::common::ActionBookmark::initializeSons:common::ActionBookmarkFolder::isChecka  
 (C++ function), 54, 627, 1134, 1236 (C++ function), 55, 628, 1135, 1237  
 sh::actions::common::ActionBookmark::isCheckableIcons:common::ActionBookmarkFolder::isChecke

(C++ function), 55, 628, 1135, 1237  
 sh::actions::common::ActionBookmarkFolder sh::actions::common::ActionBookmarks::isInitialized  
 (C++ function), 56, 629, 1136, 1237 (C++ function), 57, 630, 1137, 1239  
 sh::actions::common::ActionBookmarkFolder sh::actions::common::ActionBookmarks::isInitializing  
 (C++ function), 56, 629, 1136, 1237 (C++ function), 57, 630, 1137, 1239  
 sh::actions::common::ActionBookmarkFolder sh::actions::common::ActionBookmarks::parentAction  
 (C++ function), 56, 629, 1136, 1237 (C++ function), 57, 630, 1137, 1239  
 sh::actions::common::ActionBookmarkFolder sh::actions::common::ActionBookmarks::setChecked  
 (C++ function), 55, 628, 1135, 1237 (C++ function), 57, 630, 1137, 1238  
 sh::actions::common::ActionBookmarkFolder sh::actions::common::ActionBookmarks::setEnabled  
 (C++ function), 55, 628, 1135, 1237 (C++ function), 57, 630, 1137, 1238  
 sh::actions::common::ActionBookmarkFolder sh::actions::common::ActionBookmarks::setIcon  
 (C++ function), 55, 628, 1135, 1237 (C++ function), 57, 630, 1137, 1238  
 sh::actions::common::ActionBookmarkFolder sh::actions::common::ActionBookmarks::setSubitems  
 (C++ function), 55, 628, 1135, 1236 (C++ function), 56, 629, 1136, 1238  
 sh::actions::common::ActionBookmarkFolder sh::actions::common::ActionBookmarks::setText  
 (C++ function), 55, 628, 1135, 1237 (C++ function), 57, 630, 1137, 1238  
 sh::actions::common::ActionBookmarkFolder sh::actions::common::ActionBookmarks::setVisible  
 (C++ function), 55, 628, 1135, 1237 (C++ function), 57, 630, 1137, 1238  
 sh::actions::common::ActionBookmarkFolder sh::actions::common::ActionBookmarks::subitems  
 (C++ function), 55, 628, 1135, 1236 (C++ function), 56, 629, 1136, 1238  
 sh::actions::common::ActionBookmarkFolder sh::actions::common::ActionBookmarks::subitemsChanged  
 (C++ function), 56, 629, 1136, 1238 (C++ function), 58, 630, 1138, 1239  
 sh::actions::common::ActionBookmarkFolder sh::actions::common::ActionBookmarks::text  
 (C++ function), 55, 628, 1135, 1236 (C++ function), 56, 629, 1136, 1238  
 sh::actions::common::ActionBookmarkFolder sh::actions::common::ActionBookmarks::visible  
 (C++ function), 56, 628, 1136, 1237 (C++ function), 57, 630, 1137, 1239  
 sh::actions::common::ActionBookmarks sh::actions::common::ActionClipboardCopy  
 (C++ class), 56, 629, 1136, 1238 (C++ class), 58, 630, 1138, 1239  
 sh::actions::common::ActionBookmarks::\_setParentAction sh::actions::common::ActionClipboardCopy::\_setParentAction  
 (C++ function), 57, 630, 1137, 1239 (C++ function), 59, 632, 1139, 1240  
 sh::actions::common::ActionBookmarks::ActionClipboardCopy sh::actions::common::ActionClipboardCopy::ActionClipboardCopy  
 (C++ function), 56, 629, 1136, 1238 (C++ function), 58, 631, 1138, 1239  
 sh::actions::common::ActionBookmarks::addSubitems sh::actions::common::ActionClipboardCopy::changed  
 (C++ function), 56, 629, 1136, 1238 (C++ function), 59, 632, 1139, 1241  
 sh::actions::common::ActionBookmarks::changed sh::actions::common::ActionClipboardCopy::defaultAction  
 (C++ function), 58, 630, 1138, 1239 (C++ function), 58, 631, 1138, 1240  
 sh::actions::common::ActionBookmarks::clearSubitems sh::actions::common::ActionClipboardCopy::doInitial  
 (C++ function), 56, 629, 1136, 1238 (C++ function), 59, 632, 1139, 1241  
 sh::actions::common::ActionBookmarks::defaultAction sh::actions::common::ActionClipboardCopy::doShutdown  
 (C++ function), 57, 630, 1137, 1238 (C++ function), 59, 632, 1139, 1241  
 sh::actions::common::ActionBookmarks::enabled sh::actions::common::ActionClipboardCopy::enabled  
 (C++ function), 57, 629, 1136, 1238 (C++ function), 58, 631, 1138, 1239  
 sh::actions::common::ActionBookmarks::icon sh::actions::common::ActionClipboardCopy::execute  
 (C++ function), 56, 629, 1136, 1238 (C++ function), 58, 631, 1138, 1239  
 sh::actions::common::ActionBookmarks::initialize sh::actions::common::ActionClipboardCopy::icon  
 (C++ function), 56, 629, 1136, 1238 (C++ function), 58, 631, 1138, 1239  
 sh::actions::common::ActionBookmarks::initializeAsync sh::actions::common::ActionClipboardCopy::initializ  
 (C++ function), 57, 630, 1137, 1239 (C++ function), 59, 632, 1139, 1240  
 sh::actions::common::ActionBookmarks::initializeSync sh::actions::common::ActionClipboardCopy::initializ  
 (C++ function), 57, 630, 1137, 1239 (C++ function), 59, 632, 1139, 1240  
 sh::actions::common::ActionBookmarks::isChecked sh::actions::common::ActionClipboardCopy::isChecked  
 (C++ function), 57, 630, 1137, 1238 (C++ function), 58, 631, 1138, 1240  
 sh::actions::common::ActionBookmarks::isChecked sh::actions::common::ActionClipboardCopy::isChecked



(C++ function), 58, 631, 1138, 1240  
 sh::actions::common::ActionClipboardCopysh::actions::common::ActionClipboardCut::isInitiali-  
 (C++ function), 59, 632, 1139, 1240  
 sh::actions::common::ActionClipboardCopysh::actions::common::ActionClipboardCut::isInitiali-  
 (C++ function), 59, 632, 1139, 1240  
 sh::actions::common::ActionClipboardCopysh::actions::common::ActionClipboardCut::parentActi-  
 (C++ function), 59, 631, 1139, 1240  
 sh::actions::common::ActionClipboardCopysh::actions::common::ActionClipboardCut::setChecked-  
 (C++ function), 59, 631, 1138, 1240  
 sh::actions::common::ActionClipboardCopysh::actions::common::ActionClipboardCut::setEnabled-  
 (C++ function), 59, 631, 1138, 1240  
 sh::actions::common::ActionClipboardCopysh::actions::common::ActionClipboardCut::setIcon-  
 (C++ function), 58, 631, 1138, 1240  
 sh::actions::common::ActionClipboardCopysh::actions::common::ActionClipboardCut::setShortcu-  
 (C++ function), 58, 631, 1138, 1239  
 sh::actions::common::ActionClipboardCopysh::actions::common::ActionClipboardCut::setText-  
 (C++ function), 58, 631, 1138, 1240  
 sh::actions::common::ActionClipboardCopysh::actions::common::ActionClipboardCut::setVisible-  
 (C++ function), 59, 631, 1139, 1240  
 sh::actions::common::ActionClipboardCopysh::actions::common::ActionClipboardCut::shortcuth-  
 (C++ function), 58, 631, 1138, 1239  
 sh::actions::common::ActionClipboardCopysh::actions::common::ActionClipboardCut::shortcuth-  
 (C++ function), 58, 631, 1138, 1239  
 sh::actions::common::ActionClipboardCopysh::actions::common::ActionClipboardCut::text  
 (C++ function), 58, 631, 1138, 1239  
 sh::actions::common::ActionClipboardCopysh::actions::common::ActionClipboardCut::visible  
 (C++ function), 59, 631, 1139, 1240  
 sh::actions::common::ActionClipboardCut sh::actions::common::ActionClipboardPaste  
 (C++ class), 59, 632, 1139, 1241  
 sh::actions::common::ActionClipboardCut::sh::actions::common::ActionClipboardPaste::\_setPare-  
 (C++ function), 60, 633, 1140, 1242  
 sh::actions::common::ActionClipboardCut::sh::actions::common::ActionClipboardPaste::ActionC-  
 (C++ function), 60, 632, 1139, 1241  
 sh::actions::common::ActionClipboardCut::sh::actions::common::ActionClipboardPaste::changed  
 (C++ function), 61, 634, 1141, 1242  
 sh::actions::common::ActionClipboardCut::sh::actions::common::ActionClipboardPaste::default-  
 (C++ function), 60, 633, 1140, 1241  
 sh::actions::common::ActionClipboardCut::sh::actions::common::ActionClipboardPaste::doInitia-  
 (C++ function), 61, 634, 1141, 1242  
 sh::actions::common::ActionClipboardCut::sh::actions::common::ActionClipboardPaste::doShutdo-  
 (C++ function), 61, 634, 1141, 1242  
 sh::actions::common::ActionClipboardCut::sh::actions::common::ActionClipboardPaste::enabled  
 (C++ function), 60, 632, 1140, 1241  
 sh::actions::common::ActionClipboardCut::sh::actions::common::ActionClipboardPaste::execute  
 (C++ function), 60, 632, 1139, 1241  
 sh::actions::common::ActionClipboardCut::sh::actions::common::ActionClipboardPaste::icon  
 (C++ function), 60, 632, 1140, 1241  
 sh::actions::common::ActionClipboardCut::sh::actions::common::ActionClipboardPaste::initiali-  
 (C++ function), 61, 633, 1140, 1242  
 sh::actions::common::ActionClipboardCut::sh::actions::common::ActionClipboardPaste::initiali-  
 (C++ function), 61, 633, 1140, 1242  
 sh::actions::common::ActionClipboardCut::sh::actions::common::ActionClipboardPaste::isChecke-  
 (C++ function), 60, 633, 1140, 1241  
 sh::actions::common::ActionClipboardCut::sh::actions::common::ActionClipboardPaste::isChecke-

```

(C++ function), 62, 634, 1141, 1243
sh::actions::common::ActionClipboardPasteAsLink::isNormalizedCommon::ActionClipboardPasteAsLink::isNormalized
(C++ function), 62, 635, 1142, 1243
(C++ function), 63, 636, 1143, 1244
sh::actions::common::ActionClipboardPasteAsLink::isParentActionCommon::ActionClipboardPasteAsLink::isParentAction
(C++ function), 62, 635, 1142, 1243
(C++ function), 64, 637, 1143, 1245
sh::actions::common::ActionClipboardPasteAsLink::isEnabledCommon::ActionClipboardPasteAsLink::isEnabled
(C++ function), 62, 635, 1142, 1243
(C++ function), 64, 636, 1143, 1245
sh::actions::common::ActionClipboardPasteAsLink::setIconsCommon::ActionClipboardPasteAsLink::setIcons
(C++ function), 62, 635, 1141, 1243
(C++ function), 64, 636, 1143, 1245
sh::actions::common::ActionClipboardPasteAsLink::setShowScrollbarCommon::ActionClipboardPasteAsLink::setShowScrollbar
(C++ function), 61, 634, 1141, 1243
(C++ function), 64, 636, 1143, 1245
sh::actions::common::ActionClipboardPasteAsLink::setFlagsCommon::ActionClipboardPasteAsLink::setFlags
(C++ function), 62, 634, 1141, 1243
(C++ function), 63, 636, 1143, 1244
sh::actions::common::ActionClipboardPasteAsLink::setVisibleCommon::ActionClipboardPasteAsLink::setVisible
(C++ function), 62, 635, 1142, 1243
(C++ function), 63, 636, 1143, 1245
sh::actions::common::ActionClipboardPasteAsLink::shortNameCommon::ActionClipboardPasteAsLink::shortName
(C++ function), 61, 634, 1141, 1242
(C++ function), 64, 636, 1143, 1245
sh::actions::common::ActionClipboardPasteAsLink::shortNameOfOriginalClipboardPasteAsLink::shortNameOfOriginalClipboardPasteAsLink
(C++ function), 61, 634, 1141, 1243
(C++ function), 63, 636, 1142, 1244
sh::actions::common::ActionClipboardPasteAsLink::shortNameOfFolderClipboardPasteAsLink::shortNameOfFolderClipboardPasteAsLink
(C++ function), 61, 634, 1141, 1242
(C++ function), 63, 636, 1142, 1244
sh::actions::common::ActionClipboardPasteAsLink::textCommon::ActionClipboardPasteAsLink::text
(C++ function), 61, 634, 1141, 1243
(C++ function), 63, 636, 1143, 1244
sh::actions::common::ActionClipboardPasteAsLink::visibleCommon::ActionClipboardPasteAsLink::visible
(C++ function), 62, 635, 1142, 1243
(C++ function), 64, 636, 1143, 1245
sh::actions::common::ActionClipboardPasteAsLink::actionsCommon::ActionCopyTree
(C++ class), 63, 635, 1142, 1244
(C++ class), 64, 637, 1144, 1245
sh::actions::common::ActionClipboardPasteAsLink::setParentActionCommon::ActionCopyTree::_setParentAction
(C++ function), 64, 637, 1143, 1245
(C++ function), 66, 638, 1145, 1247
sh::actions::common::ActionClipboardPasteAsLink::actionCommon::ActionCopyTree::action
(C++ function), 63, 636, 1142, 1244
(C++ function), 65, 637, 1144, 1246
sh::actions::common::ActionClipboardPasteAsLink::changedCommon::ActionCopyTree::ActionCopyTree
(C++ function), 64, 637, 1144, 1245
(C++ function), 65, 637, 1144, 1246
sh::actions::common::ActionClipboardPasteAsLink::defaultActionCommon::ActionCopyTree::changed
(C++ function), 63, 636, 1143, 1244
(C++ function), 66, 639, 1145, 1247
sh::actions::common::ActionClipboardPasteAsLink::defaultActionCommon::ActionCopyTree::defaultAction
(C++ function), 64, 637, 1144, 1245
(C++ function), 65, 638, 1144, 1246
sh::actions::common::ActionClipboardPasteAsLink::shownCommon::ActionCopyTree::enabled
(C++ function), 64, 637, 1144, 1245
(C++ function), 65, 638, 1144, 1246
sh::actions::common::ActionClipboardPasteAsLink::executeCommon::ActionCopyTree::execute
(C++ function), 63, 636, 1143, 1244
(C++ function), 65, 637, 1144, 1246
sh::actions::common::ActionClipboardPasteAsLink::executeCommon::ActionCopyTree::icon
(C++ function), 63, 636, 1142, 1244
(C++ function), 65, 638, 1144, 1246
sh::actions::common::ActionClipboardPasteAsLink::initializeAsyncCommon::ActionCopyTree::initializeAsync
(C++ function), 63, 636, 1143, 1244
(C++ function), 66, 638, 1145, 1247
sh::actions::common::ActionClipboardPasteAsLink::initializeSyncCommon::ActionCopyTree::initializeSync
(C++ function), 64, 637, 1143, 1245
(C++ function), 66, 638, 1145, 1247
sh::actions::common::ActionClipboardPasteAsLink::isCheckedCommon::ActionCopyTree::isChecked
(C++ function), 64, 637, 1143, 1245
(C++ function), 65, 638, 1144, 1246
sh::actions::common::ActionClipboardPasteAsLink::isCheckedCommon::ActionCopyTree::isChecked
(C++ function), 64, 637, 1143, 1245
(C++ function), 65, 638, 1144, 1246

```

Index 1899

(C++ function), 68, 641, 1148, 1249  
 sh::actions::common::ActionCreateFolder:sh::actions::common::ActionDeleteItems::isChecked  
 (C++ function), 68, 641, 1147, 1249 (C++ function), 70, 642, 1149, 1251  
 sh::actions::common::ActionCreateFolder:sh::actions::common::ActionDeleteItems::isChecked  
 (C++ function), 69, 641, 1148, 1250 (C++ function), 70, 642, 1149, 1251  
 sh::actions::common::ActionCreateFolder:sh::actions::common::ActionDeleteItems::isInitiali  
 (C++ function), 69, 641, 1148, 1250 (C++ function), 71, 643, 1150, 1251  
 sh::actions::common::ActionCreateFolder:sh::actions::common::ActionDeleteItems::isInitiali  
 (C++ function), 69, 641, 1148, 1250 (C++ function), 71, 643, 1150, 1251  
 sh::actions::common::ActionCreateFolder:sh::actions::common::ActionDeleteItems::parentActi  
 (C++ function), 69, 641, 1148, 1249 (C++ function), 71, 643, 1150, 1251  
 sh::actions::common::ActionCreateFolder:sh::actions::common::ActionDeleteItems::setChecked  
 (C++ function), 69, 641, 1148, 1249 (C++ function), 70, 643, 1149, 1251  
 sh::actions::common::ActionCreateFolder:sh::actions::common::ActionDeleteItems::setEnabled  
 (C++ function), 69, 641, 1148, 1249 (C++ function), 70, 643, 1149, 1251  
 sh::actions::common::ActionCreateFolder:sh::actions::common::ActionDeleteItems::setIcon  
 (C++ function), 68, 641, 1147, 1249 (C++ function), 70, 643, 1149, 1251  
 sh::actions::common::ActionCreateFolder:sh::actions::common::ActionDeleteItems::setShortcut  
 (C++ function), 69, 641, 1148, 1249 (C++ function), 70, 642, 1149, 1250  
 sh::actions::common::ActionCreateFolder:sh::actions::common::ActionDeleteItems::setText  
 (C++ function), 69, 641, 1148, 1250 (C++ function), 70, 642, 1149, 1251  
 sh::actions::common::ActionCreateFolder:sh::actions::common::ActionDeleteItems::setVisible  
 (C++ function), 68, 640, 1147, 1249 (C++ function), 70, 643, 1149, 1251  
 sh::actions::common::ActionCreateFolder:sh::actions::common::ActionDeleteItems::shortcut  
 (C++ function), 68, 640, 1147, 1249 (C++ function), 70, 642, 1149, 1250  
 sh::actions::common::ActionCreateFolder:sh::actions::common::ActionDeleteItems::shortcut  
 (C++ function), 68, 640, 1147, 1249 (C++ function), 70, 642, 1149, 1250  
 sh::actions::common::ActionCreateFolder:sh::actions::common::ActionDeleteItems::text  
 (C++ function), 68, 641, 1147, 1249 (C++ function), 70, 642, 1149, 1251  
 sh::actions::common::ActionCreateFolder:sh::actions::common::ActionDeleteItems::visible  
 (C++ function), 69, 641, 1148, 1250 (C++ function), 70, 643, 1150, 1251  
 sh::actions::common::ActionDeleteItems sh::actions::common::ActionExecute (C++  
 (C++ class), 70, 642, 1149, 1250 class), 71, 643, 1150, 1252  
 sh::actions::common::ActionDeleteItems::sh::actions::common::ActionExecute::\_setParentActi  
 (C++ function), 71, 643, 1150, 1251 (C++ function), 72, 644, 1151, 1253  
 sh::actions::common::ActionDeleteItems::sh::actions::common::ActionExecute::ActionExecute  
 (C++ function), 70, 642, 1149, 1250 (C++ function), 71, 644, 1150, 1252  
 sh::actions::common::ActionDeleteItems::sh::actions::common::ActionExecute::changed  
 (C++ function), 71, 643, 1150, 1252 (C++ function), 73, 645, 1152, 1253  
 sh::actions::common::ActionDeleteItems::sh::actions::common::ActionExecute::defaultActionPr  
 (C++ function), 70, 642, 1149, 1251 (C++ function), 72, 644, 1151, 1252  
 sh::actions::common::ActionDeleteItems::sh::actions::common::ActionExecute::doInitialize  
 (C++ function), 71, 643, 1150, 1252 (C++ function), 73, 645, 1152, 1253  
 sh::actions::common::ActionDeleteItems::sh::actions::common::ActionExecute::doShutdown  
 (C++ function), 71, 643, 1150, 1252 (C++ function), 73, 645, 1152, 1253  
 sh::actions::common::ActionDeleteItems::sh::actions::common::ActionExecute::enabled  
 (C++ function), 70, 642, 1149, 1251 (C++ function), 72, 644, 1151, 1252  
 sh::actions::common::ActionDeleteItems::sh::actions::common::ActionExecute::execute  
 (C++ function), 70, 642, 1149, 1250 (C++ function), 71, 644, 1150, 1252  
 sh::actions::common::ActionDeleteItems::sh::actions::common::ActionExecute::icon  
 (C++ function), 70, 642, 1149, 1251 (C++ function), 72, 644, 1151, 1252  
 sh::actions::common::ActionDeleteItems::sh::actions::common::ActionExecute::initializeAsyn  
 (C++ function), 71, 643, 1150, 1251 (C++ function), 72, 645, 1151, 1253  
 sh::actions::common::ActionDeleteItems::sh::actions::common::ActionExecute::initializeSync



(C++ function), 72, 645, 1151, 1253  
 sh::actions::common::ActionExecute::isChecked (C++ function), 72, 644, 1151, 1252  
 sh::actions::common::ActionExecute::isChecked (C++ function), 72, 644, 1151, 1252  
 sh::actions::common::ActionExecute::isChecked (C++ function), 72, 645, 1151, 1253  
 sh::actions::common::ActionExecute::isChecked (C++ function), 72, 645, 1151, 1253  
 sh::actions::common::ActionExecute::parentAction (C++ function), 72, 644, 1151, 1252  
 sh::actions::common::ActionExecute::parentAction (C++ function), 73, 645, 1152, 1253  
 sh::actions::common::ActionExecute::setChecked (C++ function), 72, 645, 1151, 1253  
 sh::actions::common::ActionExecute::setChecked (C++ function), 73, 646, 1152, 1254  
 sh::actions::common::ActionExecute::setEnabled (C++ function), 72, 645, 1151, 1253  
 sh::actions::common::ActionExecute::setEnabled (C++ function), 73, 646, 1152, 1254  
 sh::actions::common::ActionExecute::setGroupActions (C++ function), 72, 644, 1151, 1253  
 sh::actions::common::ActionExecute::setGroupActions (C++ function), 73, 645, 1152, 1253  
 sh::actions::common::ActionExecute::setIcon (C++ function), 72, 644, 1151, 1253  
 sh::actions::common::ActionExecute::setIcon (C++ function), 73, 646, 1152, 1254  
 sh::actions::common::ActionExecute::setSubitems (C++ function), 72, 644, 1151, 1253  
 sh::actions::common::ActionExecute::setSubitems (C++ function), 73, 645, 1152, 1253  
 sh::actions::common::ActionExecute::setText (C++ function), 72, 644, 1151, 1252  
 sh::actions::common::ActionExecute::setText (C++ function), 73, 646, 1152, 1254  
 sh::actions::common::ActionExecute::setVisible (C++ function), 71, 644, 1150, 1252  
 sh::actions::common::ActionExecute::setVisible (C++ function), 74, 646, 1153, 1254  
 sh::actions::common::ActionExecute::subitems (C++ function), 72, 644, 1151, 1252  
 sh::actions::common::ActionExecute::subitems (C++ function), 73, 645, 1152, 1253  
 sh::actions::common::ActionExecute::subitemsChanged (C++ function), 72, 644, 1151, 1253  
 sh::actions::common::ActionExecute::subitemsChanged (C++ function), 74, 646, 1153, 1255  
 sh::actions::common::ActionExecute::text (C++ function), 71, 644, 1150, 1252  
 sh::actions::common::ActionExecute::text (C++ function), 73, 645, 1152, 1254  
 sh::actions::common::ActionExecute::triggered (C++ function), 71, 644, 1150, 1252  
 sh::actions::common::ActionExecute::triggered (C++ function), 74, 646, 1153, 1254  
 sh::actions::common::ActionExecute::textsh::actions::common::ActionHistoryNavigate (C++ function), 71, 644, 1150, 1252  
 sh::actions::common::ActionExecute::textsh::actions::common::ActionHistoryNavigate (C++ class), 74, 646, 1153, 1255  
 sh::actions::common::ActionExecute::visible (C++ function), 72, 644, 1151, 1253  
 sh::actions::common::ActionExecute::visible (C++ function), 75, 647, 1154, 1256  
 sh::actions::common::ActionGroups (C++ class), 73, 645, 1152, 1253  
 sh::actions::common::ActionGroups (C++ function), 74, 647, 1153, 1255  
 sh::actions::common::ActionGroups::\_setParentAction (C++ function), 74, 646, 1153, 1254  
 sh::actions::common::ActionGroups::\_setParentAction (C++ function), 76, 648, 1155, 1256  
 sh::actions::common::ActionGroups::ActionGroups (C++ function), 73, 645, 1152, 1253  
 sh::actions::common::ActionGroups::ActionGroups (C++ function), 75, 647, 1154, 1255  
 sh::actions::common::ActionGroups::changed (C++ function), 74, 646, 1153, 1255  
 sh::actions::common::ActionGroups::changed (C++ function), 75, 647, 1153, 1255  
 sh::actions::common::ActionGroups::defaultAction (C++ function), 73, 646, 1152, 1254  
 sh::actions::common::ActionGroups::defaultAction (C++ function), 74, 647, 1153, 1255  
 sh::actions::common::ActionGroups::enabled (C++ function), 73, 645, 1152, 1254  
 sh::actions::common::ActionGroups::enabled (C++ function), 75, 647, 1154, 1256  
 sh::actions::common::ActionGroups::icon (C++ function), 73, 645, 1152, 1254  
 sh::actions::common::ActionGroups::icon (C++ function), 75, 647, 1154, 1256  
 sh::actions::common::ActionGroups::initialize (C++ function), 74, 646, 1153, 1254  
 sh::actions::common::ActionGroups::initialize (C++ function), 75, 647, 1154, 1255  
 sh::actions::common::ActionGroups::initialize (C++ function), 74, 646, 1153, 1254  
 sh::actions::common::ActionGroups::initialize (C++ function), 75, 647, 1154, 1255  
 sh::actions::common::ActionGroups::isChecked (C++ function), 73, 645, 1152, 1254  
 sh::actions::common::ActionGroups::isChecked (C++ function), 75, 648, 1154, 1256  
 sh::actions::common::ActionGroups::isChecked (C++ function), 73, 645, 1152, 1254  
 sh::actions::common::ActionGroups::isChecked (C++ function), 75, 648, 1154, 1256  
 sh::actions::common::ActionGroups::isInitialized (C++ function), 73, 645, 1152, 1254  
 sh::actions::common::ActionGroups::isInitialized (C++ function), 75, 648, 1154, 1256  
 sh::actions::common::ActionGroups::parentAction (C++ function), 73, 645, 1152, 1253



(C++ function), 78, 650, 1157, 1258  
 sh::actions::common::ActionHistoryNavigationForwardnessSubItems::ActionManageBookmarks::text  
 (C++ function), 77, 649, 1156, 1258  
 sh::actions::common::ActionHistoryNavigationForwardnessSubItemsChangedManageBookmarks::visible  
 (C++ function), 78, 650, 1157, 1259  
 sh::actions::common::ActionHistoryNavigationForwardnessSubItemsCommon::ActionMoveTree  
 (C++ function), 77, 649, 1156, 1258  
 sh::actions::common::ActionHistoryNavigationForwardnessSubItemsCommon::ActionMoveTree::\_setParentAct  
 (C++ function), 78, 650, 1157, 1258  
 sh::actions::common::ActionManageBookmarks::actions::common::ActionMoveTree::action  
 (C++ class), 79, 650, 1157, 1259  
 sh::actions::common::ActionManageBookmarks::setParentAction::ActionMoveTree::ActionMoveTree  
 (C++ function), 80, 651, 1158, 1260  
 sh::actions::common::ActionManageBookmarks::ActionManageBookmarksOnMoveTree::changed  
 (C++ function), 79, 651, 1158, 1259  
 sh::actions::common::ActionManageBookmarks::changed::common::ActionMoveTree::defaultAction  
 (C++ function), 80, 652, 1159, 1260  
 sh::actions::common::ActionManageBookmarks::defaultActionOnMoveTree::enabled  
 (C++ function), 79, 651, 1158, 1259  
 sh::actions::common::ActionManageBookmarks::enabled::common::ActionMoveTree::execute  
 (C++ function), 79, 651, 1158, 1259  
 sh::actions::common::ActionManageBookmarks::execute::common::ActionMoveTree::icon  
 (C++ function), 79, 651, 1158, 1259  
 sh::actions::common::ActionManageBookmarks::actions::common::ActionMoveTree::initializeAsyn  
 (C++ function), 79, 651, 1158, 1259  
 sh::actions::common::ActionManageBookmarks::initializeAsyn::ActionMoveTree::initializeSyn  
 (C++ function), 80, 652, 1159, 1260  
 sh::actions::common::ActionManageBookmarks::initializeSyn::ActionMoveTree::isCheckedable  
 (C++ function), 80, 652, 1159, 1260  
 sh::actions::common::ActionManageBookmarks::isCheckedable::common::ActionMoveTree::isChecked  
 (C++ function), 79, 651, 1158, 1259  
 sh::actions::common::ActionManageBookmarks::isChecked::common::ActionMoveTree::isInitialized  
 (C++ function), 79, 651, 1158, 1259  
 sh::actions::common::ActionManageBookmarks::isInitializing::common::ActionMoveTree::isInitializing  
 (C++ function), 80, 652, 1159, 1260  
 sh::actions::common::ActionManageBookmarks::isInitializing::common::ActionMoveTree::makereadableun  
 (C++ function), 80, 652, 1159, 1260  
 sh::actions::common::ActionManageBookmarks::parentAction::common::ActionMoveTree::parentAction  
 (C++ function), 80, 651, 1158, 1260  
 sh::actions::common::ActionManageBookmarks::setChecked::common::ActionMoveTree::setChecked  
 (C++ function), 79, 651, 1158, 1260  
 sh::actions::common::ActionManageBookmarks::setEnabled::common::ActionMoveTree::setEnabled  
 (C++ function), 79, 651, 1158, 1260  
 sh::actions::common::ActionManageBookmarks::actions::common::ActionMoveTree::setIcon  
 (C++ function), 79, 651, 1158, 1260  
 sh::actions::common::ActionManageBookmarks::setShortcut::common::ActionMoveTree::setShortcut  
 (C++ function), 79, 651, 1158, 1259  
 sh::actions::common::ActionManageBookmarks::setText::common::ActionMoveTree::setText  
 (C++ function), 79, 651, 1158, 1259  
 sh::actions::common::ActionManageBookmarks::setVisible::common::ActionMoveTree::setVisible  
 (C++ function), 79, 651, 1158, 1260  
 sh::actions::common::ActionManageBookmarks::shortcut::common::ActionMoveTree::shortcut  
 (C++ function), 79, 651, 1158, 1259  
 sh::actions::common::ActionManageBookmarks::shortcut::common::ActionMoveTree::shortcut  
 (C++ function), 80, 652, 1159, 1260

(C++ function), 80, 652, 1159, 1260  
sh::actions::common::ActionMoveTree::textsh::actions::common::ActionOpenArchive  
(C++ function), 80, 652, 1159, 1261 (C++ class), 83, 655, 1162, 1263  
sh::actions::common::ActionMoveTree::visibleactions::common::ActionOpenArchive::\_setParentA  
(C++ function), 81, 653, 1160, 1261 (C++ function), 84, 656, 1163, 1264  
sh::actions::common::ActionNavigateInHistsh::actions::common::ActionOpenArchive::ActionOpenA  
(C++ class), 82, 654, 1160, 1262 (C++ function), 84, 655, 1162, 1263  
sh::actions::common::ActionNavigateInHistsh::actions::common::ActionOpenArchive::changed  
(C++ function), 83, 655, 1161, 1263 (C++ function), 85, 657, 1163, 1265  
sh::actions::common::ActionNavigateInHistsh::actions::common::ActionOpenArchive::defaultAct  
(C++ function), 82, 654, 1161, 1262 (C++ function), 84, 656, 1162, 1264  
sh::actions::common::ActionNavigateInHistsh::actions::common::ActionOpenArchive::enabled  
(C++ function), 83, 655, 1162, 1263 (C++ function), 84, 656, 1162, 1264  
sh::actions::common::ActionNavigateInHistsh::actions::common::ActionOpenArchive::execute  
(C++ function), 82, 654, 1161, 1262 (C++ function), 84, 655, 1162, 1263  
sh::actions::common::ActionNavigateInHistsh::actions::common::ActionOpenArchive::icon  
(C++ function), 82, 654, 1161, 1262 (C++ function), 84, 655, 1162, 1264  
sh::actions::common::ActionNavigateInHistsh::actions::common::ActionOpenArchive::initialized  
(C++ function), 82, 654, 1161, 1262 (C++ function), 85, 656, 1163, 1264  
sh::actions::common::ActionNavigateInHistsh::actions::common::ActionOpenArchive::initialized  
(C++ function), 82, 654, 1161, 1262 (C++ function), 85, 656, 1163, 1264  
sh::actions::common::ActionNavigateInHistsh::actions::common::ActionOpenArchive::isCheckedable  
(C++ function), 83, 655, 1162, 1263 (C++ function), 84, 656, 1162, 1264  
sh::actions::common::ActionNavigateInHistsh::actions::common::ActionOpenArchive::isCheckeded  
(C++ function), 83, 655, 1162, 1263 (C++ function), 84, 656, 1162, 1264  
sh::actions::common::ActionNavigateInHistsh::actions::common::ActionOpenArchive::isInitializ  
(C++ function), 82, 654, 1161, 1262 (C++ function), 85, 656, 1163, 1264  
sh::actions::common::ActionNavigateInHistsh::actions::common::ActionOpenArchive::isInitializ  
(C++ function), 82, 654, 1161, 1262 (C++ function), 85, 656, 1163, 1264  
sh::actions::common::ActionNavigateInHistsh::actions::common::ActionOpenArchive::parentActio  
(C++ function), 83, 655, 1162, 1263 (C++ function), 84, 656, 1163, 1264  
sh::actions::common::ActionNavigateInHistsh::actions::common::ActionOpenArchive::setChecked  
(C++ function), 83, 655, 1162, 1263 (C++ function), 84, 656, 1163, 1264  
sh::actions::common::ActionNavigateInHistsh::actions::common::ActionOpenArchive::setEnabled  
(C++ function), 83, 655, 1161, 1263 (C++ function), 84, 656, 1163, 1264  
sh::actions::common::ActionNavigateInHistsh::actions::common::ActionOpenArchive::setIcon  
(C++ function), 83, 654, 1161, 1263 (C++ function), 84, 656, 1163, 1264  
sh::actions::common::ActionNavigateInHistsh::actions::common::ActionOpenArchive::setShortcut  
(C++ function), 83, 654, 1161, 1263 (C++ function), 84, 655, 1162, 1264  
sh::actions::common::ActionNavigateInHistsh::actions::common::ActionOpenArchive::setText  
(C++ function), 83, 654, 1161, 1263 (C++ function), 84, 656, 1163, 1264  
sh::actions::common::ActionNavigateInHistsh::actions::common::ActionOpenArchive::setVisible  
(C++ function), 82, 654, 1161, 1262 (C++ function), 84, 656, 1163, 1264  
sh::actions::common::ActionNavigateInHistsh::actions::common::ActionOpenArchive::shortcutthir  
(C++ function), 82, 654, 1161, 1262 (C++ function), 84, 655, 1162, 1263  
sh::actions::common::ActionNavigateInHistsh::actions::common::ActionOpenArchive::shortcutthir  
(C++ function), 83, 655, 1161, 1263 (C++ function), 84, 655, 1162, 1263  
sh::actions::common::ActionNavigateInHistsh::actions::common::ActionOpenArchive::text  
(C++ function), 82, 654, 1161, 1262 (C++ function), 84, 655, 1162, 1264  
sh::actions::common::ActionNavigateInHistsh::actions::common::ActionOpenArchive::triggerisopenenhdirectvisible  
(C++ function), 82, 654, 1161, 1262 (C++ function), 84, 656, 1163, 1264  
sh::actions::common::ActionNavigateInHistsh::actions::common::ActionOpenDirectoryInNewWindow  
(C++ function), 82, 654, 1161, 1262 (C++ class), 85, 657, 1163, 1265  
sh::actions::common::ActionNavigateInHistsh::actions::common::ActionOpenDirectoryInNewWindow



(C++ function), 86, 658, 1164, 1266  
 sh::actions::common::ActionOpenDirectoryShNewWindow::ActionOpenDirectoryShNewWindowOpenFile  
 (C++ function), 85, 657, 1164, 1265  
 sh::actions::common::ActionOpenDirectoryShNewWindow::changed:ActionOpenFile::changed  
 (C++ function), 86, 658, 1165, 1266  
 sh::actions::common::ActionOpenDirectoryShNewWindow::defaultActionOpenFile::defaultActionOpenFile  
 (C++ function), 86, 657, 1164, 1265  
 sh::actions::common::ActionOpenDirectoryShNewWindow::doInitialActionOpenFile::doInitialize  
 (C++ function), 87, 658, 1165, 1266  
 sh::actions::common::ActionOpenDirectoryShNewWindow::doShutdownActionOpenFile::doShutdown  
 (C++ function), 87, 658, 1165, 1266  
 sh::actions::common::ActionOpenDirectoryShNewWindow::enabled:ActionOpenFile::enabled  
 (C++ function), 85, 657, 1164, 1265  
 sh::actions::common::ActionOpenDirectoryShNewWindow::execute:ActionOpenFile::execute  
 (C++ function), 85, 657, 1164, 1265  
 sh::actions::common::ActionOpenDirectoryShNewWindow::common:ActionOpenFile::icon  
 (C++ function), 85, 657, 1164, 1265  
 sh::actions::common::ActionOpenDirectoryShNewWindow::initializeAsyncOpenFile::initializeAsync  
 (C++ function), 86, 658, 1165, 1266  
 sh::actions::common::ActionOpenDirectoryShNewWindow::initializeSyncOpenFile::initializeSync  
 (C++ function), 86, 658, 1165, 1266  
 sh::actions::common::ActionOpenDirectoryShNewWindow::isChecked:ActionOpenFile::isChecked  
 (C++ function), 85, 657, 1164, 1265  
 sh::actions::common::ActionOpenDirectoryShNewWindow::isChecked:ActionOpenFile::isChecked  
 (C++ function), 85, 657, 1164, 1265  
 sh::actions::common::ActionOpenDirectoryShNewWindow::isInitialActionOpenFile::isInitialized  
 (C++ function), 86, 658, 1165, 1266  
 sh::actions::common::ActionOpenDirectoryShNewWindow::isInitialActionOpenFile::isInitializing  
 (C++ function), 86, 658, 1165, 1266  
 sh::actions::common::ActionOpenDirectoryShNewWindow::parentActionOpenFile::parentAction  
 (C++ function), 86, 658, 1164, 1266  
 sh::actions::common::ActionOpenDirectoryShNewWindow::setChecked:ActionOpenFile::setChecked  
 (C++ function), 86, 657, 1164, 1266  
 sh::actions::common::ActionOpenDirectoryShNewWindow::setEnabled:ActionOpenFile::setEnabled  
 (C++ function), 86, 657, 1164, 1265  
 sh::actions::common::ActionOpenDirectoryShNewWindow::setIcon:ActionOpenFile::setIcon  
 (C++ function), 86, 657, 1164, 1265  
 sh::actions::common::ActionOpenDirectoryShNewWindow::setShortcutActionOpenFile::setShortcut  
 (C++ function), 85, 657, 1164, 1265  
 sh::actions::common::ActionOpenDirectoryShNewWindow::setText:ActionOpenFile::setText  
 (C++ function), 86, 657, 1164, 1265  
 sh::actions::common::ActionOpenDirectoryShNewWindow::setVisible:ActionOpenFile::setVisible  
 (C++ function), 86, 658, 1164, 1266  
 sh::actions::common::ActionOpenDirectoryShNewWindow::shortcutActionOpenFile::shortcut  
 (C++ function), 85, 657, 1164, 1265  
 sh::actions::common::ActionOpenDirectoryShNewWindow::shortcutActionOpenFile::shortcut  
 (C++ function), 85, 657, 1164, 1265  
 sh::actions::common::ActionOpenDirectoryShNewWindow::text:ActionOpenFile::text  
 (C++ function), 85, 657, 1164, 1265  
 sh::actions::common::ActionOpenDirectoryShNewWindow::visible:ActionOpenFile::visible  
 (C++ function), 86, 658, 1164, 1266  
 sh::actions::common::ActionOpenFile sh::actions::common::ActionOpenFileWith  
 (C++ class), 87, 658, 1165, 1266  
 sh::actions::common::ActionOpenFile::\_setParentAction sh::actions::common::ActionOpenFileWith::\_ActionOpen

[illegible]

(C++ function), 92, 96, 663, 1170, 1271

sh::actions::common::ActionOpenFileWith:shActionOpenFileWith::setVisible  
(C++ function), 92, 96, 663, 1170, 1271

sh::actions::common::ActionOpenFileWith:shActionOpenFileWith::subitems  
(C++ function), 93, 96, 664, 1171, 1272

sh::actions::common::ActionOpenFileWith:shseParentsActioncommon::ActionOpenFileWith::subitemsCl  
(C++ function), 89, 661, 1167, 1269

sh::actions::common::ActionOpenFileWith:shActionOpenFileWith::ActionOpenFileWith::text  
(C++ function), 88, 660, 1167, 1268

sh::actions::common::ActionOpenFileWith:shhaagedons::common::ActionOpenFileWith::visible  
(C++ function), 90, 661, 1168, 1269

sh::actions::common::ActionOpenFileWith:sdefaultActionmodeActionOpenFileWith::waitProgra  
(C++ function), 89, 660, 1167, 1268

sh::actions::common::ActionOpenFileWith:shdoActionsizecommon::ActionOpenSharcArchive  
(C++ function), 90, 661, 1168, 1269

sh::actions::common::ActionOpenFileWith:shdoShutdows::common::ActionOpenSharcArchive::\_setPa  
(C++ function), 90, 661, 1168, 1269

sh::actions::common::ActionOpenFileWith:shnabledons::common::ActionOpenSharcArchive::Action  
(C++ function), 89, 660, 1167, 1268

sh::actions::common::ActionOpenFileWith:shcoactions::common::ActionOpenSharcArchive::change  
(C++ function), 89, 660, 1167, 1268

sh::actions::common::ActionOpenFileWith:shniacaloneAsyncommon::ActionOpenSharcArchive::default  
(C++ function), 89, 661, 1168, 1269

sh::actions::common::ActionOpenFileWith:shniacaloneSyncommon::ActionOpenSharcArchive::doInit  
(C++ function), 89, 661, 1168, 1269

sh::actions::common::ActionOpenFileWith:shsChetkable:common::ActionOpenSharcArchive::doShut  
(C++ function), 89, 660, 1167, 1268

sh::actions::common::ActionOpenFileWith:shsChetkeds::common::ActionOpenSharcArchive::enable  
(C++ function), 89, 660, 1167, 1268

sh::actions::common::ActionOpenFileWith:shsIactionizedommon::ActionOpenSharcArchive::execut  
(C++ function), 89, 661, 1168, 1269

sh::actions::common::ActionOpenFileWith:shsIactionizingommon::ActionOpenSharcArchive::icon  
(C++ function), 89, 661, 1168, 1269

sh::actions::common::ActionOpenFileWith:sparentActiononcommon::ActionOpenSharcArchive::initia  
(C++ function), 89, 661, 1167, 1269

sh::actions::common::ActionOpenFileWith:sRememberedOpenAmbitionActionOpenSharcArchive::initia  
(C++ class), 93

sh::actions::common::ActionOpenFileWith:sRememberedOpenAmbitionAcarguopensSharcArchive::isChec  
(C++ member), 94

sh::actions::common::ActionOpenFileWith:sRememberedOpenAmbitionActionopenSharcArchive::isChec  
(C++ member), 94

sh::actions::common::ActionOpenFileWith:sRememberedOpenAmbitionAcopenopensSharcArchive::isInit  
(C++ function), 93

sh::actions::common::ActionOpenFileWith:sRememberedOpenAmbitionAcRememberShapeAAction::isInit  
(C++ function), 93

sh::actions::common::ActionOpenFileWith:shetchecked::common::ActionOpenSharcArchive::parent  
(C++ function), 89, 660, 1167, 1268

sh::actions::common::ActionOpenFileWith:shetEnabled::common::ActionOpenSharcArchive::setChe  
(C++ function), 89, 660, 1167, 1268

sh::actions::common::ActionOpenFileWith:shetActons::common::ActionOpenSharcArchive::setEna  
(C++ function), 89, 660, 1167, 1268

sh::actions::common::ActionOpenFileWith:shetSubitems::common::ActionOpenSharcArchive::setIco  
(C++ function), 88, 660, 1167, 1268

sh::actions::common::ActionOpenFileWith:shetTexions::common::ActionOpenSharcArchive::setSho

[illegible]



(C++ function), 100, 668, 1175, 1276  
 sh::actions::common::ActionOpenTerminalAsRootActionItems::common::ActionRenameItem  
 (C++ function), 101, 668, 1175, 1276 (C++ class), 103, 671, 1177, 1279  
 sh::actions::common::ActionOpenTerminalAsRootActionItems::common::ActionRenameItem::\_setParentAction  
 (C++ function), 101, 669, 1175, 1277 (C++ function), 104, 672, 1178, 1280  
 sh::actions::common::ActionOpenTerminalAsRootActionItems::common::ActionRenameItem::ActionRenameItem  
 (C++ class), 102, 669, 1176, 1277 (C++ function), 103, 671, 1178, 1279  
 sh::actions::common::ActionOpenTerminalAsRootActionItems::common::ActionRenameItem::changed  
 (C++ function), 103, 670, 1177, 1278 (C++ function), 105, 672, 1179, 1280  
 sh::actions::common::ActionOpenTerminalAsRootActionItems::common::ActionRenameItem::defaultAction  
 (C++ function), 102, 669, 1176, 1277 (C++ function), 104, 671, 1178, 1279  
 sh::actions::common::ActionOpenTerminalAsRootActionItems::common::ActionRenameItem::doInitialize  
 (C++ function), 103, 671, 1177, 1279 (C++ function), 105, 672, 1179, 1280  
 sh::actions::common::ActionOpenTerminalAsRootActionItems::common::ActionRenameItem::doShutdown  
 (C++ function), 102, 670, 1176, 1278 (C++ function), 105, 672, 1179, 1280  
 sh::actions::common::ActionOpenTerminalAsRootActionItems::common::ActionRenameItem::enabled  
 (C++ function), 102, 670, 1176, 1278 (C++ function), 104, 671, 1178, 1279  
 sh::actions::common::ActionOpenTerminalAsRootActionItems::common::ActionRenameItem::execute  
 (C++ function), 102, 669, 1176, 1277 (C++ function), 103, 671, 1178, 1279  
 sh::actions::common::ActionOpenTerminalAsRootActionItems::common::ActionRenameItem::icon  
 (C++ function), 102, 670, 1176, 1277 (C++ function), 104, 671, 1178, 1279  
 sh::actions::common::ActionOpenTerminalAsRootActionItems::common::ActionRenameItem::initializeAsynchronous  
 (C++ function), 103, 670, 1177, 1278 (C++ function), 104, 672, 1179, 1280  
 sh::actions::common::ActionOpenTerminalAsRootActionItems::common::ActionRenameItem::initializeSynchronous  
 (C++ function), 103, 670, 1177, 1278 (C++ function), 104, 672, 1179, 1280  
 sh::actions::common::ActionOpenTerminalAsRootActionItems::common::ActionRenameItem::isChecked  
 (C++ function), 102, 670, 1176, 1278 (C++ function), 104, 671, 1178, 1279  
 sh::actions::common::ActionOpenTerminalAsRootActionItems::common::ActionRenameItem::isChecked  
 (C++ function), 102, 670, 1176, 1278 (C++ function), 104, 671, 1178, 1279  
 sh::actions::common::ActionOpenTerminalAsRootActionItems::common::ActionRenameItem::isInitialized  
 (C++ function), 103, 670, 1177, 1278 (C++ function), 104, 672, 1179, 1280  
 sh::actions::common::ActionOpenTerminalAsRootActionItems::common::ActionRenameItem::isInitializing  
 (C++ function), 103, 670, 1177, 1278 (C++ function), 104, 672, 1179, 1280  
 sh::actions::common::ActionOpenTerminalAsRootActionItems::common::ActionRenameItem::parentAction  
 (C++ function), 103, 670, 1177, 1278 (C++ function), 104, 672, 1178, 1280  
 sh::actions::common::ActionOpenTerminalAsRootActionItems::common::ActionRenameItem::setChecked  
 (C++ function), 103, 670, 1177, 1278 (C++ function), 104, 671, 1178, 1279  
 sh::actions::common::ActionOpenTerminalAsRootActionItems::common::ActionRenameItem::setEnabled  
 (C++ function), 102, 670, 1177, 1278 (C++ function), 104, 671, 1178, 1279  
 sh::actions::common::ActionOpenTerminalAsRootActionItems::common::ActionRenameItem::setIcon  
 (C++ function), 102, 670, 1177, 1278 (C++ function), 104, 671, 1178, 1279  
 sh::actions::common::ActionOpenTerminalAsRootActionItems::common::ActionRenameItem::setShortcut  
 (C++ function), 102, 669, 1176, 1277 (C++ function), 103, 671, 1178, 1279  
 sh::actions::common::ActionOpenTerminalAsRootActionItems::common::ActionRenameItem::setText  
 (C++ function), 102, 670, 1177, 1278 (C++ function), 104, 671, 1178, 1279  
 sh::actions::common::ActionOpenTerminalAsRootActionItems::common::ActionRenameItem::setVisible  
 (C++ function), 103, 670, 1177, 1278 (C++ function), 104, 672, 1178, 1279  
 sh::actions::common::ActionOpenTerminalAsRootActionItems::common::ActionRenameItem::shortcutHint  
 (C++ function), 102, 669, 1176, 1277 (C++ function), 103, 671, 1178, 1279  
 sh::actions::common::ActionOpenTerminalAsRootActionItems::common::ActionRenameItem::shortcutHint  
 (C++ function), 102, 669, 1176, 1277 (C++ function), 103, 671, 1178, 1279  
 sh::actions::common::ActionOpenTerminalAsRootActionItems::common::ActionRenameItem::text  
 (C++ function), 102, 669, 1176, 1277 (C++ function), 103, 671, 1178, 1279  
 sh::actions::common::ActionOpenTerminalAsRootActionItems::common::ActionRenameItem::visible

(C++ function), 104, 672, 1178, 1280  
 sh::actions::common::ActionShowPropertiesh::actions::common::ActionTransferToCopy  
 (C++ class), 105, 672, 1179, 1280 (C++ class), 106, 674, 1181, 1282  
 sh::actions::common::ActionShowPropertiesh::setParentActionsh::actions::common::ActionTransferToCopy::\_setParent  
 (C++ function), 106, 673, 1180, 1281 (C++ function), 107, 675, 1181, 1283  
 sh::actions::common::ActionShowPropertiesh::ActionShowPropertiesh::ActionTransferToCopy::ActionTransferToCopy  
 (C++ function), 105, 672, 1179, 1280 (C++ function), 107, 674, 1181, 1282  
 sh::actions::common::ActionShowPropertiesh::changedsh::actions::common::ActionTransferToCopy::changed  
 (C++ function), 106, 674, 1180, 1282 (C++ function), 108, 675, 1182, 1283  
 sh::actions::common::ActionShowPropertiesh::defaultActionsh::actions::common::ActionTransferToCopy::defaultAction  
 (C++ function), 105, 673, 1180, 1281 (C++ function), 107, 674, 1181, 1282  
 sh::actions::common::ActionShowPropertiesh::doInitializesh::actions::common::ActionTransferToCopy::doInitializa  
 (C++ function), 106, 674, 1181, 1282 (C++ function), 108, 675, 1182, 1283  
 sh::actions::common::ActionShowPropertiesh::doShutdownsh::actions::common::ActionTransferToCopy::doShutdown  
 (C++ function), 106, 674, 1181, 1282 (C++ function), 108, 675, 1182, 1283  
 sh::actions::common::ActionShowPropertiesh::enabledsh::actions::common::ActionTransferToCopy::enabled  
 (C++ function), 105, 673, 1179, 1281 (C++ function), 107, 674, 1181, 1282  
 sh::actions::common::ActionShowPropertiesh::setIconsh::actions::common::ActionTransferToCopy::setIcon  
 (C++ function), 105, 672, 1179, 1280 (C++ function), 107, 674, 1181, 1282  
 sh::actions::common::ActionShowPropertiesh::initializesh::actions::common::ActionTransferToCopy::initializa  
 (C++ function), 105, 673, 1179, 1281 (C++ function), 107, 675, 1182, 1283  
 sh::actions::common::ActionShowPropertiesh::initialSizeAsynch::actions::common::ActionTransferToCopy::initialSizeAsyn  
 (C++ function), 106, 673, 1180, 1281 (C++ function), 107, 675, 1182, 1283  
 sh::actions::common::ActionShowPropertiesh::initialSizeSynch::actions::common::ActionTransferToCopy::initialSizeSyn  
 (C++ function), 106, 673, 1180, 1281 (C++ function), 107, 674, 1181, 1282  
 sh::actions::common::ActionShowPropertiesh::isCheckedsh::actions::common::ActionTransferToCopy::isChecked  
 (C++ function), 105, 673, 1180, 1281 (C++ function), 107, 674, 1181, 1282  
 sh::actions::common::ActionShowPropertiesh::isInitializesh::actions::common::ActionTransferToCopy::isInitializa  
 (C++ function), 105, 673, 1179, 1281 (C++ function), 107, 675, 1182, 1283  
 sh::actions::common::ActionShowPropertiesh::isInitializesh::actions::common::ActionTransferToCopy::isInitializa  
 (C++ function), 106, 673, 1180, 1281 (C++ function), 108, 675, 1182, 1283  
 sh::actions::common::ActionShowPropertiesh::isInitializesh::actions::common::ActionTransferToCopy::parentAction  
 (C++ function), 106, 673, 1180, 1281 (C++ function), 107, 675, 1181, 1283  
 sh::actions::common::ActionShowPropertiesh::parentActionsh::actions::common::ActionTransferToCopy::setCheck  
 (C++ function), 106, 673, 1180, 1281 (C++ function), 107, 674, 1181, 1282  
 sh::actions::common::ActionShowPropertiesh::setCheckedsh::actions::common::ActionTransferToCopy::setEnabled  
 (C++ function), 106, 673, 1180, 1281 (C++ function), 107, 674, 1181, 1282  
 sh::actions::common::ActionShowPropertiesh::setEnabledsh::actions::common::ActionTransferToCopy::setIcon  
 (C++ function), 106, 673, 1180, 1281 (C++ function), 107, 674, 1181, 1282  
 sh::actions::common::ActionShowPropertiesh::setIconssh::actions::common::ActionTransferToCopy::setSubit  
 (C++ function), 106, 673, 1180, 1281 (C++ function), 107, 674, 1181, 1282  
 sh::actions::common::ActionShowPropertiesh::setShortCutsh::actions::common::ActionTransferToCopy::setText  
 (C++ function), 105, 672, 1179, 1280 (C++ function), 107, 674, 1181, 1282  
 sh::actions::common::ActionShowPropertiesh::setTexsh::actions::common::ActionTransferToCopy::setVisible  
 (C++ function), 105, 673, 1180, 1281 (C++ function), 107, 674, 1181, 1282  
 sh::actions::common::ActionShowPropertiesh::setVisiblesh::actions::common::ActionTransferToCopy::subitem  
 (C++ function), 106, 673, 1180, 1281 (C++ function), 107, 674, 1181, 1282  
 sh::actions::common::ActionShowPropertiesh::shortCutsh::actions::common::ActionTransferToCopy::subitem  
 (C++ function), 105, 672, 1179, 1280 (C++ function), 108, 675, 1182, 1283  
 sh::actions::common::ActionShowPropertiesh::shortCutsh::actions::common::ActionTransferToCopy::text  
 (C++ function), 105, 672, 1179, 1280 (C++ function), 107, 674, 1181, 1282  
 sh::actions::common::ActionShowPropertiesh::textsh::actions::common::ActionTransferToCopy::visible  
 (C++ function), 105, 673, 1179, 1280 (C++ function), 107, 675, 1181, 1282  
 sh::actions::common::ActionShowPropertiesh::visiblesh::actions::common::ActionTransferToHelper

(C++ class), 108, 675, 1182, 1283  
 sh::actions::common::ActionTransferToHelper:actions::common::ActionTransferToMove::changed  
 (C++ function), 109, 676, 1183, 1284  
 sh::actions::common::ActionTransferToHelper:actions::common::ActionTransferToMove::default  
 (C++ function), 108, 675, 1182, 1283  
 sh::actions::common::ActionTransferToHelper:actions::common::ActionTransferToMove::doInitial  
 (C++ function), 109, 677, 1184, 1285  
 sh::actions::common::ActionTransferToHelper:actions::common::ActionTransferToMove::doShutd  
 (C++ function), 109, 676, 1183, 1284  
 sh::actions::common::ActionTransferToHelper:actions::common::ActionTransferToMove::enabled  
 (C++ function), 108, 676, 1183, 1284  
 sh::actions::common::ActionTransferToHelper:actions::common::ActionTransferToMove::icon  
 (C++ function), 108, 675, 1182, 1283  
 sh::actions::common::ActionTransferToHelper:actions::common::ActionTransferToMove::initial  
 (C++ function), 108, 676, 1182, 1284  
 sh::actions::common::ActionTransferToHelper:actions::common::ActionTransferToMove::initial  
 (C++ function), 109, 676, 1183, 1284  
 sh::actions::common::ActionTransferToHelper:actions::common::ActionTransferToMove::isCheck  
 (C++ function), 109, 676, 1183, 1284  
 sh::actions::common::ActionTransferToHelper:actions::common::ActionTransferToMove::isCheck  
 (C++ function), 109, 676, 1183, 1284  
 sh::actions::common::ActionTransferToHelper:actions::common::ActionTransferToMove::isInitia  
 (C++ function), 108, 676, 1183, 1284  
 sh::actions::common::ActionTransferToHelper:actions::common::ActionTransferToMove::isInitia  
 (C++ function), 109, 676, 1183, 1284  
 sh::actions::common::ActionTransferToHelper:actions::common::ActionTransferToMove::parentAc  
 (C++ function), 109, 676, 1183, 1284  
 sh::actions::common::ActionTransferToHelper:actions::common::ActionTransferToMove::setCheck  
 (C++ function), 109, 676, 1183, 1284  
 sh::actions::common::ActionTransferToHelper:actions::common::ActionTransferToMove::setEnabl  
 (C++ function), 109, 676, 1183, 1284  
 sh::actions::common::ActionTransferToHelper:actions::common::ActionTransferToMove::setIcon  
 (C++ function), 109, 676, 1183, 1284  
 sh::actions::common::ActionTransferToHelper:actions::common::ActionTransferToMove::setSubit  
 (C++ function), 109, 676, 1183, 1284  
 sh::actions::common::ActionTransferToHelper:actions::common::ActionTransferToMove::setText  
 (C++ function), 108, 676, 1182, 1283  
 sh::actions::common::ActionTransferToHelper:actions::common::ActionTransferToMove::setVisib  
 (C++ function), 109, 676, 1183, 1284  
 sh::actions::common::ActionTransferToHelper:actions::common::ActionTransferToMove::subitem  
 (C++ function), 109, 676, 1183, 1284  
 sh::actions::common::ActionTransferToHelper:actions::common::ActionTransferToMove::subitem  
 (C++ function), 108, 675, 1182, 1283  
 sh::actions::common::ActionTransferToHelper:actions::common::ActionTransferToMove::subitem  
 (C++ function), 108, 675, 1182, 1283  
 sh::actions::common::ActionTransferToHelper:actions::common::ActionTransferToMove::visible  
 (C++ function), 108, 676, 1182, 1284  
 sh::actions::common::ActionTransferToHelper:actions::DontResolveLinksPredicate  
 (C++ function), 109, 676, 1183, 1284  
 sh::actions::common::ActionTransferToMove:sh::actions::DontResolveLinksPredicate::DontResolve  
 (C++ class), 110, 677, 1184, 1285  
 sh::actions::common::ActionTransferToMove:sh::actions::DontResolveLinksPredicate::evaluate  
 (C++ function), 110, 678, 1184, 1285  
 sh::actions::common::ActionTransferToMove:sh::actions::DontResolveLinksPredicate::prepare

<a href="#">(C++ function), 39, 614, 1121</a>	<a href="#">(C++ function), 41, 615, 1123</a>
<a href="#">sh::actions::HeaderActionItem (C++ class), 39, 614, 1121</a>	<a href="#">sh::actions::HideOnSelectionLevelPredicate::HideOnSelectionLevelPredicate (C++ function), 41, 615, 1123</a>
<a href="#">sh::actions::HeaderActionItem::_setParents (C++ function), 40, 615, 1122</a>	<a href="#">sh::actions::HideOnSelectionLevelPredicate::prepare (C++ function), 41, 615, 1123</a>
<a href="#">sh::actions::HeaderActionItem::changed (C++ function), 41, 615, 1122</a>	<a href="#">sh::actions::KeyShortcutPredicate (C++ class), 41, 615, 1123</a>
<a href="#">sh::actions::HeaderActionItem::defaultAction (C++ function), 40, 614, 1121</a>	<a href="#">sh::actions::KeyShortcutPredicate::evaluate (C++ function), 42, 616, 1123</a>
<a href="#">sh::actions::HeaderActionItem::enabled (C++ function), 40, 614, 1121</a>	<a href="#">sh::actions::KeyShortcutPredicate::KeyShortcutPredicate (C++ function), 42, 616, 1123</a>
<a href="#">sh::actions::HeaderActionItem::HeaderActionItem (C++ function), 40, 614, 1121</a>	<a href="#">sh::actions::KeyShortcutPredicate::prepare (C++ function), 42, 616, 1123</a>
<a href="#">sh::actions::HeaderActionItem::icon (C++ function), 40, 614, 1121</a>	<a href="#">sh::actions::KeyShortcutPredicate::shortcut (C++ member), 42, 616, 1123</a>
<a href="#">sh::actions::HeaderActionItem::initializeAsync (C++ function), 40, 615, 1122</a>	<a href="#">sh::actions::KeyShortcutPredicate::triggersOnCurrentAction (C++ member), 42, 616, 1123</a>
<a href="#">sh::actions::HeaderActionItem::initializeSync (C++ function), 40, 615, 1122</a>	<a href="#">sh::actions::mainmenu (C++ type), 678, 1185, 1286</a>
<a href="#">sh::actions::HeaderActionItem::isChecked (C++ function), 40, 614, 1121</a>	<a href="#">sh::actions::mainmenu::ActionAbout (C++ class), 111, 678, 1185, 1286</a>
<a href="#">sh::actions::HeaderActionItem::isChecked (C++ function), 40, 614, 1121</a>	<a href="#">sh::actions::mainmenu::ActionAbout::_setParentAction (C++ function), 112, 679, 1186, 1287</a>
<a href="#">sh::actions::HeaderActionItem::isInitiallyChecked (C++ function), 40, 615, 1122</a>	<a href="#">sh::actions::mainmenu::ActionAbout::ActionAbout (C++ function), 111, 678, 1185, 1286</a>
<a href="#">sh::actions::HeaderActionItem::isInitiallyChecked (C++ function), 40, 615, 1122</a>	<a href="#">sh::actions::mainmenu::ActionAbout::changed (C++ function), 113, 680, 1187, 1288</a>
<a href="#">sh::actions::HeaderActionItem::parentAction (C++ function), 40, 614, 1122</a>	<a href="#">sh::actions::mainmenu::ActionAbout::defaultAction (C++ function), 112, 679, 1186, 1287</a>
<a href="#">sh::actions::HeaderActionItem::setChecked (C++ function), 40, 614, 1122</a>	<a href="#">sh::actions::mainmenu::ActionAbout::enabled (C++ function), 111, 679, 1186, 1287</a>
<a href="#">sh::actions::HeaderActionItem::setEnabled (C++ function), 40, 614, 1122</a>	<a href="#">sh::actions::mainmenu::ActionAbout::execute (C++ function), 111, 678, 1185, 1286</a>
<a href="#">sh::actions::HeaderActionItem::setIcon (C++ function), 40, 614, 1122</a>	<a href="#">sh::actions::mainmenu::ActionAbout::icon (C++ function), 111, 679, 1186, 1287</a>
<a href="#">sh::actions::HeaderActionItem::setText (C++ function), 40, 614, 1122</a>	<a href="#">sh::actions::mainmenu::ActionAbout::initializeAsync (C++ function), 112, 679, 1186, 1287</a>
<a href="#">sh::actions::HeaderActionItem::setVisible (C++ function), 40, 614, 1122</a>	<a href="#">sh::actions::mainmenu::ActionAbout::initializeSync (C++ function), 112, 679, 1186, 1287</a>
<a href="#">sh::actions::HeaderActionItem::text (C++ function), 40, 614, 1121</a>	<a href="#">sh::actions::mainmenu::ActionAbout::isChecked (C++ function), 112, 679, 1186, 1287</a>
<a href="#">sh::actions::HeaderActionItem::visible (C++ function), 40, 614, 1122</a>	<a href="#">sh::actions::mainmenu::ActionAbout::isChecked (C++ function), 112, 679, 1186, 1287</a>
<a href="#">sh::actions::HideOnCurrentDirectoryLevelPredicate (C++ class), 41, 615, 1122</a>	<a href="#">sh::actions::mainmenu::ActionAbout::isInitialized (C++ function), 112, 679, 1186, 1287</a>
<a href="#">sh::actions::HideOnCurrentDirectoryLevelPredicate (C++ function), 41, 615, 1122</a>	<a href="#">sh::actions::mainmenu::ActionAbout::isInitializing (C++ function), 112, 679, 1186, 1287</a>
<a href="#">sh::actions::HideOnCurrentDirectoryLevelPredicate (C++ function), 41, 615, 1122</a>	<a href="#">sh::actions::mainmenu::ActionAbout::parentAction (C++ function), 112, 679, 1186, 1287</a>
<a href="#">sh::actions::HideOnCurrentDirectoryLevelPredicate (C++ function), 41, 615, 1122</a>	<a href="#">sh::actions::mainmenu::ActionAbout::setChecked (C++ function), 112, 679, 1186, 1287</a>
<a href="#">sh::actions::HideOnSelectionLevelPredicate (C++ class), 41, 615, 1122</a>	<a href="#">sh::actions::mainmenu::ActionAbout::setEnabled (C++ function), 112, 679, 1186, 1287</a>
<a href="#">sh::actions::HideOnSelectionLevelPredicate (C++ function), 41, 615, 1123</a>	<a href="#">sh::actions::mainmenu::ActionAbout::setIcon (C++ function), 112, 679, 1186, 1287</a>



**Index** **1913**

```

sh::actions::mainmenu::ActionApplyProfileh:(C++ function), 116, 117, 683, 1190, 1291
sh::actions::mainmenu::ActionApplyProfileh:(C++ class), 118, 684, 1191, 1292
sh::actions::mainmenu::ActionApplyProfileh:(C++ function), 116, 118, 683, 1190, 1291
sh::actions::mainmenu::ActionApplyProfileh:(C++ function), 119, 685, 1192, 1293
sh::actions::mainmenu::ActionApplyProfileh:(C++ function), 116, 117, 683, 1190, 1291
sh::actions::mainmenu::ActionApplyProfileh:(C++ function), 119, 684, 1191, 1292
sh::actions::mainmenu::ActionApplyProfileh:(C++ function), 116, 117, 683, 1190, 1291
sh::actions::mainmenu::ActionApplyProfileh:(C++ function), 120, 685, 1192, 1293
sh::actions::mainmenu::ActionApplyProfileh:(C++ function), 116, 117, 683, 1190, 1291
sh::actions::mainmenu::ActionApplyProfileh:(C++ function), 119, 685, 1191, 1292
sh::actions::mainmenu::ActionApplyProfileh:(C++ function), 116, 118, 683, 1190, 1291
sh::actions::mainmenu::ActionApplyProfileh:(C++ function), 119, 684, 1191, 1292
sh::actions::mainmenu::ActionApplyProfileh:changeds::mainmenu::ActionFileViewChangeIconDimen(C++)
sh::actions::mainmenu::ActionApplyProfileh:changeds::mainmenu::ActionFileViewChangeIconDimen(C++)
sh::actions::mainmenu::ActionApplyProfileh:defaultActionmainmenu::ActionFileViewChangeIconDimen(C++)
sh::actions::mainmenu::ActionApplyProfileh:enableds::mainmenu::ActionFileViewChangeIconDimen(C++)
sh::actions::mainmenu::ActionApplyProfileh:isactions::mainmenu::ActionFileViewChangeIconDimen(C++)
sh::actions::mainmenu::ActionApplyProfileh:isactionsizesynms::mainmenu::ActionFileViewChangeIconDimen(C++)
sh::actions::mainmenu::ActionApplyProfileh:isactionsizesynms::mainmenu::ActionFileViewChangeIconDimen(C++)
sh::actions::mainmenu::ActionApplyProfileh:isCheckedabimainmenu::ActionFileViewChangeIconDimen(C++)
sh::actions::mainmenu::ActionApplyProfileh:isCheckedabimainmenu::ActionFileViewChangeIconDimen(C++)
sh::actions::mainmenu::ActionApplyProfileh:isCheckedabimainmenu::ActionFileViewChangeIconDimen(C++)
sh::actions::mainmenu::ActionApplyProfileh:isCheckedabimainmenu::ActionFileViewChangeIconDimen(C++)
sh::actions::mainmenu::ActionApplyProfileh:isdisabledmainmenu::ActionFileViewChangeIconDimen(C++)
sh::actions::mainmenu::ActionApplyProfileh:isdisabledmainmenu::ActionFileViewChangeIconDimen(C++)
sh::actions::mainmenu::ActionApplyProfileh:isdisabledmainmenu::ActionFileViewChangeIconDimen(C++)
sh::actions::mainmenu::ActionApplyProfileh:isdisabledmainmenu::ActionFileViewChangeIconDimen(C++)
sh::actions::mainmenu::ActionApplyProfileh:parentAstmainmenu::ActionFileViewChangeIconDimen(C++)
sh::actions::mainmenu::ActionApplyProfileh:setCheckedmainmenu::ActionFileViewChangeIconDimen(C++)
sh::actions::mainmenu::ActionApplyProfileh:setEnabledmainmenu::ActionFileViewChangeIconDimen(C++)
sh::actions::mainmenu::ActionApplyProfileh:setIcons::mainmenu::ActionFileViewChangeIconDimen(C++)
sh::actions::mainmenu::ActionApplyProfileh:setSubItemsmainmenu::ActionFileViewChangeIconDimen(C++)
sh::actions::mainmenu::ActionApplyProfileh:setTexts::mainmenu::ActionFileViewChangeIconDimen(C++)
sh::actions::mainmenu::ActionApplyProfileh:setVisiblemainmenu::ActionFileViewChangeIconDimen(C++)
sh::actions::mainmenu::ActionApplyProfileh:subItems::mainmenu::ActionFileViewChangeIconDimen(C++)
sh::actions::mainmenu::ActionApplyProfileh:subItemsChangemenu::ActionFileViewChangeIconDimen(C++)
sh::actions::mainmenu::ActionApplyProfileh:textions::mainmenu::ActionFileViewDecreaseIconDimen(C++)
sh::actions::mainmenu::ActionApplyProfileh:vastibbs::mainmenu::ActionFileViewDecreaseIconDimen(C++)

```

(C++ function), 121, 686, 1193, 1294  
 sh::actions::mainmenu::ActionFileViewDecshaseactionismenactAonFonhViewNewFdeeeizeFDimet  
 (C++ function), 120, 686, 1193, 1294  
 (C++ function), 123, 687, 1194, 1295  
 sh::actions::mainmenu::ActionFileViewDecshaseactionismenachaAgedonFileviewFilesizeFormat  
 (C++ function), 121, 687, 1194, 1295  
 (C++ function), 123, 687, 1194, 1295  
 sh::actions::mainmenu::ActionFileViewDecshaseactionismenendefAoffAnFidePreweddeasezeFormat  
 (C++ function), 120, 686, 1193, 1294  
 (C++ function), 123, 687, 1194, 1295  
 sh::actions::mainmenu::ActionFileViewDecshaseactionismenenaAgedonFileviewFilesizeFormat  
 (C++ function), 120, 686, 1193, 1294  
 (C++ function), 123, 687, 1194, 1295  
 sh::actions::mainmenu::ActionFileViewDecshaseactionismenexeAutéonFileviewFilesizeFormat  
 (C++ function), 120, 686, 1193, 1294  
 (C++ function), 124, 688, 1195, 1296  
 sh::actions::mainmenu::ActionFileViewDecshaseactionismenicoActionFileviewFilesizeFormat  
 (C++ function), 120, 686, 1193, 1294  
 (C++ function), 124, 688, 1195, 1296  
 sh::actions::mainmenu::ActionFileViewDecshaseactionismenuniAcalonEAsyucwFilesizeFormat  
 (C++ function), 121, 687, 1194, 1295  
 (C++ function), 123, 687, 1194, 1295  
 sh::actions::mainmenu::ActionFileViewDecshaseactionismenuniAcalonESyevviewFilesizeFormat  
 (C++ function), 121, 687, 1194, 1295  
 (C++ function), 123, 687, 1194, 1295  
 sh::actions::mainmenu::ActionFileViewDecshaseactionismenüsChetkahFéviewFilesizeFormat  
 (C++ function), 120, 686, 1193, 1294  
 (C++ function), 124, 688, 1195, 1296  
 sh::actions::mainmenu::ActionFileViewDecshaseactionismenüsChetkedFileviewFilesizeFormat  
 (C++ function), 120, 686, 1193, 1294  
 (C++ function), 124, 688, 1195, 1296  
 sh::actions::mainmenu::ActionFileViewDecshaseactionismenüsIActionFizédviewFilesizeFormat  
 (C++ function), 121, 687, 1194, 1295  
 (C++ function), 124, 688, 1195, 1296  
 sh::actions::mainmenu::ActionFileViewDecshaseactionismenüsIActionFizlegiewFilesizeFormat  
 (C++ function), 121, 687, 1194, 1295  
 (C++ function), 124, 688, 1195, 1296  
 sh::actions::mainmenu::ActionFileViewDecshaseactionismenparAntAonFonhviewFilesizeFormat  
 (C++ function), 121, 686, 1193, 1294  
 (C++ function), 124, 688, 1195, 1296  
 sh::actions::mainmenu::ActionFileViewDecshaseactionismensetAcheckedFileviewFilesizeFormat  
 (C++ function), 121, 686, 1193, 1294  
 (C++ function), 124, 688, 1195, 1296  
 sh::actions::mainmenu::ActionFileViewDecshaseactionismensetAnabbedFileviewFilesizeFormat  
 (C++ function), 121, 686, 1193, 1294  
 (C++ function), 123, 687, 1194, 1295  
 sh::actions::mainmenu::ActionFileViewDecshaseactionismensetActmonFileviewFilesizeFormat  
 (C++ function), 121, 686, 1193, 1294  
 (C++ function), 124, 688, 1195, 1296  
 sh::actions::mainmenu::ActionFileViewDecshaseactionismensetAhorbonFileviewFilesizeFormat  
 (C++ function), 120, 686, 1193, 1294  
 (C++ function), 124, 688, 1195, 1296  
 sh::actions::mainmenu::ActionFileViewDecshaseactionismensetAekfonFileviewFilesizeFormat  
 (C++ function), 121, 686, 1193, 1294  
 (C++ function), 123, 687, 1194, 1295  
 sh::actions::mainmenu::ActionFileViewDecshaseactionismensetAcsibhFileviewFilesizeFormat  
 (C++ function), 121, 686, 1193, 1294  
 (C++ function), 123, 687, 1194, 1295  
 sh::actions::mainmenu::ActionFileViewDecshaseactionismenshoAttwtohFileviewFilesizeFormat  
 (C++ function), 120, 686, 1193, 1294  
 (C++ function), 123, 687, 1194, 1295  
 sh::actions::mainmenu::ActionFileViewDecshaseactionismenshoAttwtohFileviggErisOnCueFonmD  
 (C++ function), 120, 686, 1193, 1294  
 (C++ function), 124, 688, 1195, 1296  
 sh::actions::mainmenu::ActionFileViewDecshaseactionismenutexActionFileViewIncreaseIconD  
 (C++ function), 120, 686, 1193, 1294  
 (C++ class), 121, 688, 1195, 1296  
 sh::actions::mainmenu::ActionFileViewDecshaseactionismenvisAbtéonFileViewIncreaseIconD  
 (C++ function), 121, 686, 1193, 1294  
 (C++ function), 122, 689, 1196, 1297  
 sh::actions::mainmenu::ActionFileviewFileshizeFormastimginmenu::ActionFileViewIncreaseIconD  
 (C++ class), 123, 687, 1194, 1295  
 (C++ function), 122, 689, 1196, 1297  
 sh::actions::mainmenu::ActionFileviewFileshizeFormastimginmentPakehtAnFideViewIncreaseIconD  
 (C++ function), 124, 688, 1195, 1296  
 (C++ function), 123, 690, 1197, 1298  
 sh::actions::mainmenu::ActionFileviewFileshizeFormastimginAenúonActéonFéVéewZeFoemaeFongD  
 (C++ function), 123, 687, 1194, 1295  
 (C++ function), 122, 689, 1196, 1297  
 sh::actions::mainmenu::ActionFileviewFileshizeFormastimginmchangeActionFileViewIncreaseIconD





(C++ function), 127, 693, 1200, 1301  
 sh::actions::mainmenu::ActionFileviewViewsh::isCheckablemainmenu::ActionGotoDirectory::isInitia  
 (C++ function), 126, 692, 1199, 1300 (C++ function), 129, 694, 1201, 1302  
 sh::actions::mainmenu::ActionFileviewViewsh::isCheckedmainmenu::ActionGotoDirectory::isInitia  
 (C++ function), 126, 692, 1199, 1300 (C++ function), 129, 694, 1201, 1302  
 sh::actions::mainmenu::ActionFileviewViewsh::isMinimalizedmainmenu::ActionGotoDirectory::parentA  
 (C++ function), 127, 693, 1200, 1301 (C++ function), 128, 694, 1201, 1302  
 sh::actions::mainmenu::ActionFileviewViewsh::isMinimalizingmainmenu::ActionGotoDirectory::setChe  
 (C++ function), 127, 693, 1200, 1301 (C++ function), 128, 694, 1201, 1302  
 sh::actions::mainmenu::ActionFileviewViewsh::parentActionmainmenu::ActionGotoDirectory::setEnab  
 (C++ function), 127, 692, 1199, 1300 (C++ function), 128, 694, 1201, 1302  
 sh::actions::mainmenu::ActionFileviewViewsh::setCheckedmainmenu::ActionGotoDirectory::setIcon  
 (C++ function), 127, 692, 1199, 1300 (C++ function), 128, 694, 1201, 1302  
 sh::actions::mainmenu::ActionFileviewViewsh::setEnabledmainmenu::ActionGotoDirectory::setShor  
 (C++ function), 127, 692, 1199, 1300 (C++ function), 128, 693, 1200, 1301  
 sh::actions::mainmenu::ActionFileviewViewsh::setIconsmainmenu::ActionGotoDirectory::setText  
 (C++ function), 127, 692, 1199, 1300 (C++ function), 128, 694, 1201, 1302  
 sh::actions::mainmenu::ActionFileviewViewsh::setShortcutsmainmenu::ActionGotoDirectory::setVis  
 (C++ function), 126, 692, 1199, 1300 (C++ function), 128, 694, 1201, 1302  
 sh::actions::mainmenu::ActionFileviewViewsh::setTitlesmainmenu::ActionGotoDirectory::shortcu  
 (C++ function), 127, 692, 1199, 1300 (C++ function), 128, 693, 1200, 1301  
 sh::actions::mainmenu::ActionFileviewViewsh::setVisiblemainmenu::ActionGotoDirectory::shortcu  
 (C++ function), 127, 692, 1199, 1300 (C++ function), 128, 693, 1200, 1301  
 sh::actions::mainmenu::ActionFileviewViewsh::shortcutsmainmenu::ActionGotoDirectory::text  
 (C++ function), 126, 692, 1199, 1300 (C++ function), 128, 693, 1200, 1301  
 sh::actions::mainmenu::ActionFileviewViewsh::shortcutsmainmenu::ActionGotoDirectory::visible  
 (C++ function), 126, 692, 1199, 1300 (C++ function), 128, 694, 1201, 1302  
 sh::actions::mainmenu::ActionFileviewViewsh::textsactions::mainmenu::ActionHelp (C++  
 (C++ function), 126, 692, 1199, 1300 class), 129, 694, 1201, 1302  
 sh::actions::mainmenu::ActionFileviewViewsh::visibilityactions::mainmenu::ActionHelp::\_setParentAction  
 (C++ function), 127, 692, 1199, 1300 (C++ function), 130, 695, 1202, 1303  
 sh::actions::mainmenu::ActionGotoDirectorysh::actions::mainmenu::ActionHelp::ActionHelp  
 (C++ class), 127, 693, 1200, 1301 (C++ function), 129, 695, 1202, 1302  
 sh::actions::mainmenu::ActionGotoDirectorysh::addActionmainmenu::ActionHelp::changed  
 (C++ function), 128, 694, 1201, 1302 (C++ function), 130, 696, 1203, 1304  
 sh::actions::mainmenu::ActionGotoDirectorysh::ActionGotoDirectoryActionHelp::defaultActionPre  
 (C++ function), 128, 693, 1200, 1301 (C++ function), 129, 695, 1202, 1303  
 sh::actions::mainmenu::ActionGotoDirectorysh::changedmainmenu::ActionHelp::enabled  
 (C++ function), 129, 694, 1201, 1302 (C++ function), 129, 695, 1202, 1303  
 sh::actions::mainmenu::ActionGotoDirectorysh::defaultActionmainmenu::ActionHelp::execute  
 (C++ function), 128, 693, 1200, 1301 (C++ function), 129, 695, 1202, 1302  
 sh::actions::mainmenu::ActionGotoDirectorysh::enabledmainmenu::ActionHelp::icon  
 (C++ function), 128, 693, 1200, 1301 (C++ function), 129, 695, 1202, 1303  
 sh::actions::mainmenu::ActionGotoDirectorysh::executeactions::mainmenu::ActionHelp::initializeAsync  
 (C++ function), 128, 693, 1200, 1301 (C++ function), 130, 696, 1203, 1303  
 sh::actions::mainmenu::ActionGotoDirectorysh::actions::mainmenu::ActionHelp::initializeSync  
 (C++ function), 128, 693, 1200, 1301 (C++ function), 130, 696, 1203, 1303  
 sh::actions::mainmenu::ActionGotoDirectorysh::initializeAsyncmainmenu::ActionHelp::isCheckable  
 (C++ function), 129, 694, 1201, 1302 (C++ function), 129, 695, 1202, 1303  
 sh::actions::mainmenu::ActionGotoDirectorysh::initializeSyncmainmenu::ActionHelp::isChecked  
 (C++ function), 129, 694, 1201, 1302 (C++ function), 129, 695, 1202, 1303  
 sh::actions::mainmenu::ActionGotoDirectorysh::isCheckablemainmenu::ActionHelp::isInitialized  
 (C++ function), 128, 693, 1200, 1301 (C++ function), 130, 696, 1203, 1303  
 sh::actions::mainmenu::ActionGotoDirectorysh::isCheckedmainmenu::ActionHelp::isInitializing

(C++ function), 130, 696, 1203, 1303 (C++ function), 131, 697, 1204, 1305  
sh::actions::mainmenu::ActionHelp::parentAction sh::actions::mainmenu::ActionInvertNodeSelection:::  
(C++ function), 130, 695, 1202, 1303 (C++ function), 131, 697, 1204, 1304  
sh::actions::mainmenu::ActionHelp::setChecked sh::actions::mainmenu::ActionInvertNodeSelection:::  
(C++ function), 130, 695, 1202, 1303 (C++ function), 131, 697, 1204, 1304  
sh::actions::mainmenu::ActionHelp::setEnabled sh::actions::mainmenu::ActionInvertNodeSelection:::  
(C++ function), 130, 695, 1202, 1303 (C++ function), 131, 696, 1203, 1304  
sh::actions::mainmenu::ActionHelp::setIcon sh::actions::mainmenu::ActionInvertNodeSelection:::  
(C++ function), 130, 695, 1202, 1303 (C++ function), 131, 697, 1204, 1304  
sh::actions::mainmenu::ActionHelp::setShortcathions sh::actions::mainmenu::ActionInvertNodeSelection:::  
(C++ function), 129, 695, 1202, 1303 (C++ function), 131, 697, 1204, 1305  
sh::actions::mainmenu::ActionHelp::setTex sh::actions::mainmenu::ActionInvertNodeSelection:::  
(C++ function), 130, 695, 1202, 1303 (C++ function), 131, 696, 1203, 1304  
sh::actions::mainmenu::ActionHelp::setVisible sh::actions::mainmenu::ActionInvertNodeSelection:::  
(C++ function), 130, 695, 1202, 1303 (C++ function), 131, 696, 1203, 1304  
sh::actions::mainmenu::ActionHelp::shortshthant sh::actions::mainmenu::ActionInvertNodeSelection:::  
(C++ function), 129, 695, 1202, 1302 (C++ function), 131, 696, 1203, 1304  
sh::actions::mainmenu::ActionHelp::shortshthant sh::actions::mainmenu::ActionInvertNodeSelection:::  
(C++ function), 129, 695, 1202, 1302 (C++ function), 131, 697, 1204, 1305  
sh::actions::mainmenu::ActionHelp::text sh::actions::mainmenu::ActionInvertNodeSelection:::v  
(C++ function), 129, 695, 1202, 1303 (C++ member), 132, 697, 1204, 1305  
sh::actions::mainmenu::ActionHelp::visib sh::actions::mainmenu::ActionMain (C++  
(C++ function), 130, 695, 1202, 1303 class), 132, 697, 1204, 1305  
sh::actions::mainmenu::ActionInvertNodeSelecactions sh::actions::mainmenu::ActionMain::\_setParentAction  
(C++ class), 130, 696, 1203, 1304 (C++ function), 133, 698, 1205, 1306  
sh::actions::mainmenu::ActionInvertNodeSelecactions sh::actions::mainmenu::ActionMain::ActionMain  
(C++ function), 131, 697, 1204, 1305 (C++ function), 134, 699, 1206, 1307  
sh::actions::mainmenu::ActionInvertNodeSelecactions sh::actions::mainmenu::ActionMain::ActionMain  
(C++ function), 131, 696, 1203, 1304 (C++ function), 133, 699, 1206, 1307  
sh::actions::mainmenu::ActionInvertNodeSelecactions sh::actions::mainmenu::ActionMain::defaultActionPre  
(C++ function), 132, 697, 1204, 1305 (C++ function), 132, 698, 1205, 1306  
sh::actions::mainmenu::ActionInvertNodeSelecactions sh::actions::mainmenu::ActionMain::doInitialize  
(C++ function), 131, 696, 1203, 1304 (C++ function), 133, 699, 1206, 1306  
sh::actions::mainmenu::ActionInvertNodeSelecactions sh::actions::mainmenu::ActionMain::doShutdown  
(C++ function), 131, 696, 1203, 1304 (C++ function), 133, 699, 1206, 1306  
sh::actions::mainmenu::ActionInvertNodeSelecactions sh::actions::mainmenu::ActionMain::enabled  
(C++ function), 131, 696, 1203, 1304 (C++ function), 132, 698, 1205, 1305  
sh::actions::mainmenu::ActionInvertNodeSelecactions sh::actions::mainmenu::ActionMain::icon  
(C++ function), 131, 696, 1203, 1304 (C++ function), 132, 698, 1205, 1305  
sh::actions::mainmenu::ActionInvertNodeSelecactions sh::actions::mainmenu::ActionMain::initializeAsync  
(C++ function), 132, 697, 1204, 1305 (C++ function), 133, 698, 1205, 1306  
sh::actions::mainmenu::ActionInvertNodeSelecactions sh::actions::mainmenu::ActionMain::initializeSync  
(C++ function), 132, 697, 1204, 1305 (C++ function), 133, 698, 1205, 1306  
sh::actions::mainmenu::ActionInvertNodeSelecactions sh::actions::mainmenu::ActionMain::isAlive  
(C++ function), 131, 696, 1203, 1304 (C++ function), 133, 699, 1206, 1306  
sh::actions::mainmenu::ActionInvertNodeSelecactions sh::actions::mainmenu::ActionMain::isCheckable  
(C++ function), 131, 696, 1203, 1304 (C++ function), 132, 698, 1205, 1306  
sh::actions::mainmenu::ActionInvertNodeSelecactions sh::actions::mainmenu::ActionMain::isChecked  
(C++ function), 132, 697, 1204, 1305 (C++ function), 132, 698, 1205, 1305  
sh::actions::mainmenu::ActionInvertNodeSelecactions sh::actions::mainmenu::ActionMain::isInitialized  
(C++ function), 132, 697, 1204, 1305 (C++ function), 133, 698, 1205, 1306  
sh::actions::mainmenu::ActionInvertNodeSelecactions sh::actions::mainmenu::ActionMain::isInitializing  
(C++ function), 131, 697, 1204, 1305 (C++ function), 133, 698, 1205, 1306  
sh::actions::mainmenu::ActionInvertNodeSelecactions sh::actions::mainmenu::ActionMain::parentAction

(C++ function), 133, 698, 1205, 1306  
 sh::actions::mainmenu::ActionMain::setCheckedActions::mainmenu::ActionManageSavedSettings:::  
 (C++ function), 133, 698, 1205, 1306 (C++ function), 134, 700, 1207, 1308  
 sh::actions::mainmenu::ActionMain::setEnabledActions::mainmenu::ActionManageSavedSettings:::  
 (C++ function), 133, 698, 1205, 1306 (C++ function), 134, 700, 1207, 1307  
 sh::actions::mainmenu::ActionMain::setIconsh::actions::mainmenu::ActionManageSavedSettings:::  
 (C++ function), 133, 698, 1205, 1306 (C++ function), 134, 699, 1206, 1307  
 sh::actions::mainmenu::ActionMain::setSubItemsActions::mainmenu::ActionManageSavedSettings:::  
 (C++ function), 132, 698, 1205, 1305 (C++ function), 134, 700, 1207, 1307  
 sh::actions::mainmenu::ActionMain::setTexsh::actions::mainmenu::ActionManageSavedSettings:::  
 (C++ function), 133, 698, 1205, 1306 (C++ function), 135, 700, 1207, 1308  
 sh::actions::mainmenu::ActionMain::setVisibleActions::mainmenu::ActionManageSavedSettings:::  
 (C++ function), 133, 698, 1205, 1306 (C++ function), 134, 699, 1206, 1307  
 sh::actions::mainmenu::ActionMain::shutdown::actions::mainmenu::ActionManageSavedSettings:::  
 (C++ function), 133, 699, 1206, 1306 (C++ function), 134, 699, 1206, 1307  
 sh::actions::mainmenu::ActionMain::subItems::actions::mainmenu::ActionManageSavedSettings:::  
 (C++ function), 132, 698, 1205, 1305 (C++ function), 134, 699, 1206, 1307  
 sh::actions::mainmenu::ActionMain::subItemsChangeds::mainmenu::ActionManageSavedSettings:::  
 (C++ function), 133, 699, 1206, 1307 (C++ function), 135, 700, 1207, 1308  
 sh::actions::mainmenu::ActionMain::text sh::actions::mainmenu::ActionNumberFileviews  
 (C++ function), 132, 698, 1205, 1305 (C++ class), 135, 700, 1207, 1308  
 sh::actions::mainmenu::ActionMain::visibsh::actions::mainmenu::ActionNumberFileviews::\_setI  
 (C++ function), 133, 698, 1205, 1306 (C++ function), 136, 701, 1208, 1309  
 sh::actions::mainmenu::ActionManageSavedSettings::actions::mainmenu::ActionNumberFileviews::Actio  
 (C++ class), 134, 699, 1206, 1307 (C++ function), 135, 701, 1208, 1308  
 sh::actions::mainmenu::ActionManageSavedSettings::semBarmenActActionNumberFileviews::chang  
 (C++ function), 135, 700, 1207, 1308 (C++ function), 136, 702, 1209, 1309  
 sh::actions::mainmenu::ActionManageSavedSettings::onActionManageSavedSettingsFileviews::defau  
 (C++ function), 134, 699, 1206, 1307 (C++ function), 136, 701, 1208, 1309  
 sh::actions::mainmenu::ActionManageSavedSettings::onshangedmenu::ActionNumberFileviews::enabl  
 (C++ function), 135, 700, 1207, 1308 (C++ function), 135, 701, 1208, 1308  
 sh::actions::mainmenu::ActionManageSavedSettings::ondefaultActionNumberFileviews::icon  
 (C++ function), 134, 700, 1207, 1307 (C++ function), 135, 701, 1208, 1308  
 sh::actions::mainmenu::ActionManageSavedSettings::onenabledmenu::ActionNumberFileviews::init  
 (C++ function), 134, 699, 1206, 1307 (C++ function), 136, 701, 1208, 1309  
 sh::actions::mainmenu::ActionManageSavedSettings::onexamaitemenu::ActionNumberFileviews::init  
 (C++ function), 134, 699, 1206, 1307 (C++ function), 136, 701, 1208, 1309  
 sh::actions::mainmenu::ActionManageSavedSettings::oniscmainmenu::ActionNumberFileviews::isChe  
 (C++ function), 134, 699, 1206, 1307 (C++ function), 135, 701, 1208, 1309  
 sh::actions::mainmenu::ActionManageSavedSettings::oninimainmenu::ActionNumberFileviews::isChe  
 (C++ function), 135, 700, 1207, 1308 (C++ function), 135, 701, 1208, 1308  
 sh::actions::mainmenu::ActionManageSavedSettings::oninimainmenu::SynActionNumberFileviews::isIn  
 (C++ function), 135, 700, 1207, 1308 (C++ function), 136, 701, 1208, 1309  
 sh::actions::mainmenu::ActionManageSavedSettings::oniscmainmenu::ActionNumberFileviews::isIn  
 (C++ function), 134, 700, 1207, 1307 (C++ function), 136, 702, 1209, 1309  
 sh::actions::mainmenu::ActionManageSavedSettings::oniscmainmenu::ActionNumberFileviews::paren  
 (C++ function), 134, 699, 1206, 1307 (C++ function), 136, 701, 1208, 1309  
 sh::actions::mainmenu::ActionManageSavedSettings::oniscmainmenu::ActionNumberFileviews::setCh  
 (C++ function), 135, 700, 1207, 1308 (C++ function), 136, 701, 1208, 1309  
 sh::actions::mainmenu::ActionManageSavedSettings::oniscmainmenu::ActionNumberFileviews::setEn  
 (C++ function), 135, 700, 1207, 1308 (C++ function), 136, 701, 1208, 1309  
 sh::actions::mainmenu::ActionManageSavedSettings::oniscmainmenu::ActionNumberFileviews::setI  
 (C++ function), 135, 700, 1207, 1308 (C++ function), 136, 701, 1208, 1309  
 sh::actions::mainmenu::ActionManageSavedSettings::oniscmainmenu::ActionNumberFileviews::setS

[illegible]



(C++ function), 139, 705, 1211, 1312  
 sh::actions::mainmenu::ActionNumberFileviewsRemove:sh::mainmenu::ActionNumberFileviews\_Remove  
 (C++ function), 139, 704, 1211, 1311  
 (C++ function), 140, 705, 1212, 1313  
 sh::actions::mainmenu::ActionNumberFileviewsRemove:sh::mainmenu::ActionNumberFileviews\_Remove  
 (C++ function), 139, 704, 1211, 1311  
 (C++ function), 140, 705, 1212, 1313  
 sh::actions::mainmenu::ActionNumberFileviewsRemove:sh::mainmenu::ActionNumberFileviews\_Remove  
 (C++ function), 139, 704, 1211, 1312  
 (C++ function), 140, 705, 1212, 1313  
 sh::actions::mainmenu::ActionNumberFileviewsRemove:sh::mainmenu::ActionNumberFileviews\_Remove  
 (C++ function), 139, 705, 1211, 1312  
 (C++ function), 141, 706, 1213, 1314  
 sh::actions::mainmenu::ActionNumberFileviewsRemove:sh::mainmenu::ActionQuit (C++  
 (C++ class), 140, 705, 1212, 1313  
 class), 142, 707, 1213, 1314  
 sh::actions::mainmenu::ActionNumberFileviewsRemove:sh::mainmenu::setParentAction  
 (C++ function), 141, 706, 1213, 1314  
 (C++ function), 143, 708, 1214, 1315  
 sh::actions::mainmenu::ActionNumberFileviewsRemove:sh::mainmenu::ActionNumberFileviews\_Remove  
 (C++ function), 140, 705, 1212, 1313  
 (C++ function), 142, 707, 1214, 1314  
 sh::actions::mainmenu::ActionNumberFileviewsRemove:sh::mainmenu::changed  
 (C++ function), 141, 707, 1213, 1314  
 (C++ function), 143, 708, 1215, 1316  
 sh::actions::mainmenu::ActionNumberFileviewsRemove:sh::mainmenu::ActionPredefinedActionPre  
 (C++ function), 141, 706, 1212, 1313  
 (C++ function), 142, 707, 1214, 1315  
 sh::actions::mainmenu::ActionNumberFileviewsRemove:sh::mainmenu::ActionQuit::enabled  
 (C++ function), 142, 707, 1213, 1314  
 (C++ function), 142, 707, 1214, 1315  
 sh::actions::mainmenu::ActionNumberFileviewsRemove:sh::mainmenu::ActionQuit::execute  
 (C++ function), 142, 707, 1213, 1314  
 (C++ function), 142, 707, 1214, 1314  
 sh::actions::mainmenu::ActionNumberFileviewsRemove:sh::mainmenu::ActionQuit::icon  
 (C++ function), 140, 706, 1212, 1313  
 (C++ function), 142, 707, 1214, 1315  
 sh::actions::mainmenu::ActionNumberFileviewsRemove:sh::mainmenu::ActionQuit::initializeAsync  
 (C++ function), 140, 705, 1212, 1313  
 (C++ function), 143, 708, 1215, 1315  
 sh::actions::mainmenu::ActionNumberFileviewsRemove:sh::mainmenu::ActionQuit::initializeSync  
 (C++ function), 140, 705, 1212, 1313  
 (C++ function), 143, 708, 1215, 1315  
 sh::actions::mainmenu::ActionNumberFileviewsRemove:sh::mainmenu::ActionQuit::isAsyncCheckable  
 (C++ function), 141, 706, 1213, 1314  
 (C++ function), 142, 707, 1214, 1315  
 sh::actions::mainmenu::ActionNumberFileviewsRemove:sh::mainmenu::ActionQuit::isSyncChecked  
 (C++ function), 141, 706, 1213, 1314  
 (C++ function), 142, 707, 1214, 1315  
 sh::actions::mainmenu::ActionNumberFileviewsRemove:sh::mainmenu::ActionQuit::isInitialized  
 (C++ function), 140, 706, 1212, 1313  
 (C++ function), 143, 708, 1215, 1315  
 sh::actions::mainmenu::ActionNumberFileviewsRemove:sh::mainmenu::ActionQuit::isInitializing  
 (C++ function), 140, 706, 1212, 1313  
 (C++ function), 143, 708, 1215, 1315  
 sh::actions::mainmenu::ActionNumberFileviewsRemove:sh::mainmenu::ActionQuit::parentAction  
 (C++ function), 141, 706, 1213, 1314  
 (C++ function), 143, 708, 1214, 1315  
 sh::actions::mainmenu::ActionNumberFileviewsRemove:sh::mainmenu::ActionQuit::setChecked  
 (C++ function), 141, 706, 1213, 1314  
 (C++ function), 143, 708, 1214, 1315  
 sh::actions::mainmenu::ActionNumberFileviewsRemove:sh::mainmenu::ActionQuit::setEnabled  
 (C++ function), 141, 706, 1213, 1314  
 (C++ function), 142, 708, 1214, 1315  
 sh::actions::mainmenu::ActionNumberFileviewsRemove:sh::mainmenu::ActionQuit::setIcon  
 (C++ function), 141, 706, 1213, 1313  
 (C++ function), 142, 707, 1214, 1315  
 sh::actions::mainmenu::ActionNumberFileviewsRemove:sh::mainmenu::ActionQuit::setShortcutHint  
 (C++ function), 141, 706, 1213, 1313  
 (C++ function), 142, 707, 1214, 1314  
 sh::actions::mainmenu::ActionNumberFileviewsRemove:sh::mainmenu::ActionQuit::setText  
 (C++ function), 141, 706, 1213, 1313  
 (C++ function), 142, 707, 1214, 1315  
 sh::actions::mainmenu::ActionNumberFileviewsRemove:sh::mainmenu::ActionQuit::setVisible  
 (C++ function), 140, 705, 1212, 1313  
 (C++ function), 143, 708, 1214, 1315  
 sh::actions::mainmenu::ActionNumberFileviewsRemove:sh::mainmenu::ActionQuit::shortcutHint  
 (C++ function), 141, 706, 1213, 1313  
 (C++ function), 142, 707, 1214, 1314  
 sh::actions::mainmenu::ActionNumberFileviewsRemove:sh::mainmenu::ActionQuit::shortcutHintTri

```

(C++ function), 142, 707, 1214, 1315
sh::actions::mainmenu::ActionQuit::text sh::actions::mainmenu::ActionSaveSettings
(C++ function), 142, 707, 1214, 1315
(C++ class), 145, 710, 1216, 1317
sh::actions::mainmenu::ActionQuit::visible sh::actions::mainmenu::ActionSaveSettings::_setParameter
(C++ function), 143, 708, 1214, 1315
(C++ function), 146, 711, 1217, 1318
sh::actions::mainmenu::ActionRefresh sh::actions::mainmenu::ActionSaveSettings::ActionSaveSettings
(C++ class), 143, 708, 1215, 1316
(C++ function), 145, 710, 1216, 1317
sh::actions::mainmenu::ActionRefresh::_setParameter sh::actions::mainmenu::ActionSaveSettings::changed
(C++ function), 144, 709, 1216, 1317
(C++ function), 146, 711, 1218, 1318
sh::actions::mainmenu::ActionRefresh::ActionRefresh sh::actions::mainmenu::ActionSaveSettings::defaultText
(C++ function), 143, 708, 1215, 1316
(C++ function), 145, 710, 1217, 1318
sh::actions::mainmenu::ActionRefresh::changed sh::actions::mainmenu::ActionSaveSettings::enabled
(C++ function), 145, 710, 1216, 1317
(C++ function), 145, 710, 1217, 1317
sh::actions::mainmenu::ActionRefresh::defaultText sh::actions::mainmenu::ActionSaveSettings::execute
(C++ function), 144, 709, 1215, 1316
(C++ function), 145, 710, 1216, 1317
sh::actions::mainmenu::ActionRefresh::enabled sh::actions::mainmenu::ActionSaveSettings::icon
(C++ function), 144, 709, 1215, 1316
(C++ function), 145, 710, 1217, 1317
sh::actions::mainmenu::ActionRefresh::execute sh::actions::mainmenu::ActionSaveSettings::initialText
(C++ function), 143, 708, 1215, 1316
(C++ function), 146, 711, 1217, 1318
sh::actions::mainmenu::ActionRefresh::icon sh::actions::mainmenu::ActionSaveSettings::initialText
(C++ function), 144, 709, 1215, 1316
(C++ function), 146, 711, 1217, 1318
sh::actions::mainmenu::ActionRefresh::initialText sh::actions::mainmenu::ActionSaveSettings::isChecked
(C++ function), 144, 709, 1216, 1317
(C++ function), 145, 710, 1217, 1318
sh::actions::mainmenu::ActionRefresh::initialText sh::actions::mainmenu::ActionSaveSettings::isChecked
(C++ function), 144, 709, 1216, 1317
(C++ function), 145, 710, 1217, 1317
sh::actions::mainmenu::ActionRefresh::isChecked sh::actions::mainmenu::ActionSaveSettings::isInitialText
(C++ function), 144, 709, 1215, 1316
(C++ function), 146, 711, 1217, 1318
sh::actions::mainmenu::ActionRefresh::isChecked sh::actions::mainmenu::ActionSaveSettings::isInitialText
(C++ function), 144, 709, 1215, 1316
(C++ function), 146, 711, 1217, 1318
sh::actions::mainmenu::ActionRefresh::isInitialText sh::actions::mainmenu::ActionSaveSettings::parentAction
(C++ function), 144, 709, 1216, 1317
(C++ function), 146, 711, 1217, 1318
sh::actions::mainmenu::ActionRefresh::isInitialText sh::actions::mainmenu::ActionSaveSettings::setCheck
(C++ function), 144, 709, 1216, 1317
(C++ function), 146, 710, 1217, 1318
sh::actions::mainmenu::ActionRefresh::parentAction sh::actions::mainmenu::ActionSaveSettings::setEnabled
(C++ function), 144, 709, 1216, 1317
(C++ function), 145, 710, 1217, 1318
sh::actions::mainmenu::ActionRefresh::setEnabled sh::actions::mainmenu::ActionSaveSettings::setIcon
(C++ function), 144, 709, 1216, 1316
(C++ function), 145, 710, 1217, 1318
sh::actions::mainmenu::ActionRefresh::setEnabled sh::actions::mainmenu::ActionSaveSettings::setShort
(C++ function), 144, 709, 1216, 1316
(C++ function), 145, 710, 1217, 1317
sh::actions::mainmenu::ActionRefresh::setIcon sh::actions::mainmenu::ActionSaveSettings::setText
(C++ function), 144, 709, 1216, 1316
(C++ function), 145, 710, 1217, 1318
sh::actions::mainmenu::ActionRefresh::setShortText sh::actions::mainmenu::ActionSaveSettings::setVisible
(C++ function), 143, 708, 1215, 1316
(C++ function), 146, 711, 1217, 1318
sh::actions::mainmenu::ActionRefresh::setText sh::actions::mainmenu::ActionSaveSettings::shortcut
(C++ function), 144, 709, 1216, 1316
(C++ function), 145, 710, 1216, 1317
sh::actions::mainmenu::ActionRefresh::setVisible sh::actions::mainmenu::ActionSaveSettings::shortcut
(C++ function), 144, 709, 1216, 1316
(C++ function), 145, 710, 1216, 1317
sh::actions::mainmenu::ActionRefresh::shortcut sh::actions::mainmenu::ActionSaveSettings::text
(C++ function), 143, 708, 1215, 1316
(C++ function), 145, 710, 1217, 1317
sh::actions::mainmenu::ActionRefresh::shortcut sh::actions::mainmenu::ActionSaveSettings::visible
(C++ function), 143, 708, 1215, 1316
(C++ function), 146, 711, 1217, 1318
sh::actions::mainmenu::ActionRefresh::text sh::actions::mainmenu::ActionSearch
(C++ function), 143, 708, 1215, 1316
(C++ class), 146, 711, 1218, 1318
sh::actions::mainmenu::ActionRefresh::visible sh::actions::mainmenu::ActionSearch::setParentAct
(C++ function), 143, 708, 1215, 1316
(C++ function), 146, 711, 1218, 1318

```

(C++ function), 147, 712, 1219, 1319  
 sh::actions::mainmenu::ActionSearch::ActionSearchns::mainmenu::ActionSelectAllNodes::default  
 (C++ function), 146, 711, 1218, 1319  
 sh::actions::mainmenu::ActionSearch::changedactions::mainmenu::ActionSelectAllNodes::enable  
 (C++ function), 148, 713, 1219, 1320  
 sh::actions::mainmenu::ActionSearch::defaultActionBreadCrumbs::mainmenu::ActionSelectAllNodes::execut  
 (C++ function), 147, 712, 1218, 1319  
 sh::actions::mainmenu::ActionSearch::enabledactions::mainmenu::ActionSelectAllNodes::icon  
 (C++ function), 147, 711, 1218, 1319  
 sh::actions::mainmenu::ActionSearch::executeactions::mainmenu::ActionSelectAllNodes::initia  
 (C++ function), 146, 711, 1218, 1319  
 sh::actions::mainmenu::ActionSearch::iconsh::actions::mainmenu::ActionSelectAllNodes::initia  
 (C++ function), 147, 711, 1218, 1319  
 sh::actions::mainmenu::ActionSearch::initialActionAsync::mainmenu::ActionSelectAllNodes::isChe  
 (C++ function), 147, 712, 1219, 1320  
 sh::actions::mainmenu::ActionSearch::initialActions::mainmenu::ActionSelectAllNodes::isChe  
 (C++ function), 147, 712, 1219, 1320  
 sh::actions::mainmenu::ActionSearch::isCheckedactions::mainmenu::ActionSelectAllNodes::isInit  
 (C++ function), 147, 712, 1218, 1319  
 sh::actions::mainmenu::ActionSearch::isCheckedactions::mainmenu::ActionSelectAllNodes::isInit  
 (C++ function), 147, 712, 1218, 1319  
 sh::actions::mainmenu::ActionSearch::isInitializeds::mainmenu::ActionSelectAllNodes::parent  
 (C++ function), 147, 712, 1219, 1320  
 sh::actions::mainmenu::ActionSearch::isInitializing::mainmenu::ActionSelectAllNodes::setChe  
 (C++ function), 147, 712, 1219, 1320  
 sh::actions::mainmenu::ActionSearch::parentActions::mainmenu::ActionSelectAllNodes::setEna  
 (C++ function), 147, 712, 1219, 1319  
 sh::actions::mainmenu::ActionSearch::setCheckedactions::mainmenu::ActionSelectAllNodes::setIco  
 (C++ function), 147, 712, 1219, 1319  
 sh::actions::mainmenu::ActionSearch::setEnabledactions::mainmenu::ActionSelectAllNodes::setSho  
 (C++ function), 147, 712, 1218, 1319  
 sh::actions::mainmenu::ActionSearch::setShortactions::mainmenu::ActionSelectAllNodes::setTex  
 (C++ function), 147, 712, 1218, 1319  
 sh::actions::mainmenu::ActionSearch::setShortActionInst::mainmenu::ActionSelectAllNodes::setVis  
 (C++ function), 146, 711, 1218, 1319  
 sh::actions::mainmenu::ActionSearch::setTextactions::mainmenu::ActionSelectAllNodes::shortc  
 (C++ function), 147, 712, 1218, 1319  
 sh::actions::mainmenu::ActionSearch::setVisibleactions::mainmenu::ActionSelectAllNodes::shortc  
 (C++ function), 147, 712, 1219, 1319  
 sh::actions::mainmenu::ActionSearch::showShortcuts::mainmenu::ActionSelectAllNodes::text  
 (C++ function), 146, 711, 1218, 1319  
 sh::actions::mainmenu::ActionSearch::showShortcutsInToolbar::mainmenu::ActionSelectAllNodes::visibi  
 (C++ function), 146, 711, 1218, 1319  
 sh::actions::mainmenu::ActionSearch::textsh::actions::mainmenu::ActionSetWindowTitlePattern  
 (C++ function), 146, 711, 1218, 1319  
 sh::actions::mainmenu::ActionSearch::visibleactions::mainmenu::ActionSetWindowTitlePattern  
 (C++ function), 147, 712, 1219, 1319  
 sh::actions::mainmenu::ActionSelectAllNodessh::actions::mainmenu::ActionSetWindowTitlePattern  
 (C++ class), 148, 713, 1219, 1320  
 sh::actions::mainmenu::ActionSelectAllNodessh::actions::mainmenu::ActionSetWindowTitlePattern  
 (C++ function), 149, 714, 1220, 1321  
 sh::actions::mainmenu::ActionSelectAllNodessh::actions::mainmenu::ActionSetWindowTitlePattern  
 (C++ function), 151, 716, 1222, 1323  
 sh::actions::mainmenu::ActionSelectAllNodessh::actions::mainmenu::ActionSetWindowTitlePattern  
 (C++ function), 148, 713, 1219, 1320  
 sh::actions::mainmenu::ActionSelectAllNodessh::actions::mainmenu::ActionSetWindowTitlePattern  
 (C++ function), 150, 715, 1221, 1322  
 sh::actions::mainmenu::ActionSelectAllNodessh::actions::mainmenu::ActionSetWindowTitlePattern

(C++ function), 150, 714, 1221, 1322  
 sh::actions::mainmenu::ActionSetWindowTishPatterns:enableMainMenu::ActionShowDetailPanel::init  
 (C++ function), 149, 714, 1221, 1321  
 (C++ function), 152, 717, 1223, 1324  
 sh::actions::mainmenu::ActionSetWindowTishPatterns:imainmenu::ActionShowDetailPanel::init  
 (C++ function), 150, 714, 1221, 1322  
 (C++ function), 152, 717, 1223, 1324  
 sh::actions::mainmenu::ActionSetWindowTishPatterns:imainmenu::ActionShowDetailPanel::isChecked  
 (C++ function), 150, 715, 1222, 1322  
 (C++ function), 151, 716, 1223, 1323  
 sh::actions::mainmenu::ActionSetWindowTishPatterns:imainmenu::ActionShowDetailPanel::isChecked  
 (C++ function), 150, 715, 1222, 1322  
 (C++ function), 151, 716, 1222, 1323  
 sh::actions::mainmenu::ActionSetWindowTishPatterns:isCheckableActionShowDetailPanel::isIn  
 (C++ function), 150, 715, 1221, 1322  
 (C++ function), 152, 717, 1223, 1324  
 sh::actions::mainmenu::ActionSetWindowTishPatterns:isChecked::ActionShowDetailPanel::isIn  
 (C++ function), 150, 715, 1221, 1322  
 (C++ function), 152, 717, 1223, 1324  
 sh::actions::mainmenu::ActionSetWindowTishPatterns:isImmediateActionShowDetailPanel::paren  
 (C++ function), 150, 715, 1222, 1322  
 (C++ function), 152, 717, 1223, 1324  
 sh::actions::mainmenu::ActionSetWindowTishPatterns:isImmediateActionShowDetailPanel::setCl  
 (C++ function), 150, 715, 1222, 1322  
 (C++ function), 152, 716, 1223, 1324  
 sh::actions::mainmenu::ActionSetWindowTishPatterns:parentActionShowDetailPanel::setEn  
 (C++ function), 150, 715, 1222, 1322  
 (C++ function), 151, 716, 1223, 1323  
 sh::actions::mainmenu::ActionSetWindowTishPatterns:isChecked::ActionShowDetailPanel::setI  
 (C++ function), 150, 715, 1221, 1322  
 (C++ function), 151, 716, 1223, 1323  
 sh::actions::mainmenu::ActionSetWindowTishPatterns:isEnabled::ActionShowDetailPanel::setS  
 (C++ function), 150, 715, 1221, 1322  
 (C++ function), 151, 716, 1222, 1323  
 sh::actions::mainmenu::ActionSetWindowTishPatterns:isMainMenu::ActionShowDetailPanel::setT  
 (C++ function), 150, 715, 1221, 1322  
 (C++ function), 151, 716, 1223, 1323  
 sh::actions::mainmenu::ActionSetWindowTishPatterns:isShowMenu::ActionShowDetailPanel::setV  
 (C++ function), 149, 714, 1221, 1322  
 (C++ function), 152, 717, 1223, 1324  
 sh::actions::mainmenu::ActionSetWindowTishPatterns:isMainMenu::ActionShowDetailPanel::short  
 (C++ function), 150, 715, 1221, 1322  
 (C++ function), 151, 716, 1222, 1323  
 sh::actions::mainmenu::ActionSetWindowTishPatterns:isMainMenu::ActionShowDetailPanel::short  
 (C++ function), 150, 715, 1221, 1322  
 (C++ function), 151, 716, 1222, 1323  
 sh::actions::mainmenu::ActionSetWindowTishPatterns:isMainMenu::ActionShowDetailPanel::text  
 (C++ function), 149, 714, 1221, 1321  
 (C++ function), 151, 716, 1222, 1323  
 sh::actions::mainmenu::ActionSetWindowTishPatterns:isMainMenu::ActionShowDetailPanel::text  
 (C++ function), 149, 714, 1221, 1321  
 (C++ function), 152, 717, 1223, 1324  
 sh::actions::mainmenu::ActionSetWindowTishPatterns:isMainMenu::ActionShowFolderTree  
 (C++ function), 149, 714, 1221, 1322  
 (C++ class), 152, 717, 1223, 1324  
 sh::actions::mainmenu::ActionSetWindowTishPatterns:isMainMenu::ActionShowFolderTree::\_setPa  
 (C++ function), 150, 715, 1222, 1322  
 (C++ function), 153, 718, 1224, 1325  
 sh::actions::mainmenu::ActionShowDetailPanelactions::mainmenu::ActionShowFolderTree::Action  
 (C++ class), 151, 716, 1222, 1323  
 (C++ function), 152, 717, 1224, 1324  
 sh::actions::mainmenu::ActionShowDetailPanelactions::mainmenu::ActionShowFolderTree::change  
 (C++ function), 152, 717, 1223, 1324  
 (C++ function), 154, 719, 1225, 1326  
 sh::actions::mainmenu::ActionShowDetailPanelactions::mainmenu::ActionShowFolderTree::defaul  
 (C++ function), 151, 716, 1222, 1323  
 (C++ function), 153, 718, 1224, 1325  
 sh::actions::mainmenu::ActionShowDetailPanelactions::mainmenu::ActionShowFolderTree::enabl  
 (C++ function), 152, 717, 1223, 1324  
 (C++ function), 153, 717, 1224, 1325  
 sh::actions::mainmenu::ActionShowDetailPanelactions::mainmenu::ActionShowFolderTree::execut  
 (C++ function), 151, 716, 1223, 1323  
 (C++ function), 152, 717, 1224, 1324  
 sh::actions::mainmenu::ActionShowDetailPanelactions::mainmenu::ActionShowFolderTree::icon  
 (C++ function), 151, 716, 1222, 1323  
 (C++ function), 153, 717, 1224, 1324  
 sh::actions::mainmenu::ActionShowDetailPanelactions::mainmenu::ActionShowFolderTree::initia  
 (C++ function), 151, 716, 1222, 1323  
 (C++ function), 153, 718, 1225, 1325  
 sh::actions::mainmenu::ActionShowDetailPanelactions::mainmenu::ActionShowFolderTree::initia





```
(C++ function), 156, 721, 1227, 1328  
sh::actions::mainmenu::ActionTreeStickyness:aparentActionMenu::setEnabled  
(C++ function), 156, 721, 1227, 1328  
sh::actions::mainmenu::ActionTreeStickyness:setCheckedmainmenu::ActionTuning::setIcon  
(C++ function), 156, 721, 1227, 1328  
sh::actions::mainmenu::ActionTreeStickyness:setEnabledmainmenu::ActionTuning::setShortcutIcon  
(C++ function), 156, 721, 1227, 1328  
sh::actions::mainmenu::ActionTreeStickyness:setIconmainmenu::ActionTuning::setText  
(C++ function), 156, 721, 1227, 1328  
sh::actions::mainmenu::ActionTreeStickyness:setShortCutmainmenu::ActionTuning::setVisible  
(C++ function), 155, 720, 1227, 1327  
sh::actions::mainmenu::ActionTreeStickyness:setTextmainmenu::ActionTuning::shortcutHint  
(C++ function), 156, 721, 1227, 1328  
sh::actions::mainmenu::ActionTreeStickyness:setVisiblemainmenu::ActionTuning::shortcutHintTitle  
(C++ function), 156, 721, 1227, 1328  
sh::actions::mainmenu::ActionTreeStickyness:showSubMenumainmenu::ActionTuning::text  
(C++ function), 155, 720, 1227, 1327  
sh::actions::mainmenu::ActionTreeStickyness:showSubMenumainmenu::ActionTuning::textObject  
(C++ function), 155, 720, 1227, 1327  
sh::actions::mainmenu::ActionTreeStickyness:actions::OnDirectoriesPredicate  
(C++ class), 42, 616, 1123  
sh::actions::mainmenu::ActionTreeStickyness:actions::OnDirectoriesPredicate::evaluate  
(C++ function), 42, 616, 1123  
sh::actions::mainmenu::ActionTuning sh::actions::OnDirectoriesPredicate::OnDirectoriesPredicate  
(C++ class), 157, 722, 1228, 1328  
sh::actions::mainmenu::ActionTuning::_setUpParentActionOnDirectoriesPredicate::prepare  
(C++ function), 158, 723, 1229, 1329  
sh::actions::mainmenu::ActionTuning::Actions::OnFilesPredicate(C++ class),  
(C++ function), 157, 722, 1228, 1329  
sh::actions::mainmenu::ActionTuning::changedactions::OnFilesPredicate::evaluate  
(C++ function), 158, 723, 1229, 1330  
sh::actions::mainmenu::ActionTuning::defaultActionBreakNamesPredicate::OnFilesPredicate  
(C++ function), 157, 722, 1228, 1329  
sh::actions::mainmenu::ActionTuning::enabledactions::OnFilesPredicate::prepare  
(C++ function), 157, 722, 1228, 1329  
sh::actions::mainmenu::ActionTuning::executeactions::OnLinksPredicate(C++ class),  
(C++ function), 157, 722, 1228, 1329  
sh::actions::mainmenu::ActionTuning::iconsh::actions::OnLinksPredicate::evaluate  
(C++ function), 157, 722, 1228, 1329  
sh::actions::mainmenu::ActionTuning::initializeAsync::OnLinksPredicate::OnLinksPredicate  
(C++ function), 158, 723, 1229, 1330  
sh::actions::mainmenu::ActionTuning::initializeSons::OnLinksPredicate::prepare  
(C++ function), 158, 723, 1229, 1330  
sh::actions::mainmenu::ActionTuning::isCheckedActions::OnSingleEntrySelectionPredicate  
(C++ function), 157, 722, 1228, 1329  
sh::actions::mainmenu::ActionTuning::isCheckableActions::OnSingleEntrySelectionPredicate::evaluate  
(C++ function), 157, 722, 1228, 1329  
sh::actions::mainmenu::ActionTuning::isInitialized::OnSingleEntrySelectionPredicate::OnInit  
(C++ function), 158, 723, 1229, 1330  
sh::actions::mainmenu::ActionTuning::isInitializing::OnSingleEntrySelectionPredicate::prepare  
(C++ function), 158, 723, 1229, 1330  
sh::actions::mainmenu::ActionTuning::parentActions::PositionIndexPredicate  
(C++ class), 43, 617, 1124  
sh::actions::mainmenu::ActionTuning::setCheckedActions::PositionIndexPredicate::evaluate
```

<b>Index</b>	<b>1927</b>
--------------	-------------

[illegible]

sh::base::LogSeverity::INFO (C++ <i>enumerator</i> ), 723, 1330	sh::base::ShallotProcess::logPrefix (C++ <i>function</i> ), 163, 727, 1334
sh::base::LogSeverity::WARNING (C++ <i>enumerator</i> ), 723, 1330	sh::base::ShallotProcess::maxbrowserawayseconds (C++ <i>function</i> ), 163, 728, 1335
sh::base::MainThread (C++ <i>class</i> ), 162, 726, 1333	sh::base::ShallotProcess::maxidlenessesseconds (C++ <i>function</i> ), 163, 728, 1335
sh::base::MainThread::dispatcher (C++ <i>function</i> ), 162, 727, 1334	sh::base::ShallotProcess::maxmemorypercent (C++ <i>function</i> ), 163, 728, 1335
sh::base::ShallotProcess (C++ <i>class</i> ), 162, 727, 1334	sh::base::ShallotProcess::maxmemorypercentglobal (C++ <i>function</i> ), 163, 728, 1335
sh::base::ShallotProcess::_brandingcolorsh (C++ <i>member</i> ), 164, 729, 1336	sh::base::ShallotProcess::maxnumberworkers (C++ <i>function</i> ), 163, 727, 1334
sh::base::ShallotProcess::_brandingcolorsh::base::ShallotProcess::maxnumberworkersperhost (C++ <i>function</i> ), 163, 728, 1334	sh::base::ShallotProcess::maxport (C++ <i>function</i> ), 163, 727, 1334
sh::base::ShallotProcess::_cfgvalassignmentsh (C++ <i>member</i> ), 164, 729, 1336	sh::base::ShallotProcess::minport (C++ <i>function</i> ), 163, 727, 1334
sh::base::ShallotProcess::_cfgvalfiles (C++ <i>member</i> ), 164, 729, 1336	sh::base::ShallotProcess::parameters (C++ <i>function</i> ), 163, 727, 1334
sh::base::ShallotProcess::_cmdlineargs (C++ <i>member</i> ), 164, 729, 1336	sh::base::ShallotProcess::parameterValue (C++ <i>function</i> ), 163, 727, 1334
sh::base::ShallotProcess::_maxport (C++ <i>member</i> ), 164, 729, 1336	sh::base::ShallotProcess::parameterValueInt (C++ <i>function</i> ), 163, 727, 1334
sh::base::ShallotProcess::_minport (C++ <i>member</i> ), 164, 729, 1336	sh::base::ShallotProcess::parameterValueList (C++ <i>function</i> ), 163, 727, 1334
sh::base::ShallotProcess::_ui (C++ <i>member</i> ), 164, 729, 1336	sh::base::ShallotProcess::revisionString (C++ <i>function</i> ), 164, 728, 1335
sh::base::ShallotProcess::_workdir (C++ <i>member</i> ), 164, 729, 1336	sh::base::ShallotProcess::shallotDataDir (C++ <i>function</i> ), 164, 728, 1335
sh::base::ShallotProcess::_workermaxportsh (C++ <i>member</i> ), 164, 729, 1336	sh::base::ShallotProcess::ShallotProcess (C++ <i>function</i> ), 164, 729, 1335
sh::base::ShallotProcess::_workerminportsh (C++ <i>member</i> ), 164, 729, 1336	sh::base::ShallotProcess::shutdown (C++ <i>function</i> ), 164, 728, 1335
sh::base::ShallotProcess::brandingColor (C++ <i>function</i> ), 163, 727, 1334	sh::base::ShallotProcess::uiMode (C++ <i>function</i> ), 163, 727, 1334
sh::base::ShallotProcess::buildtime (C++ <i>function</i> ), 164, 728, 1335	sh::base::ShallotProcess::userDataDir (C++ <i>function</i> ), 164, 728, 1335
sh::base::ShallotProcess::cfgvalBrandingColorsh (C++ <i>member</i> ), 164, 729, 1336	sh::base::ShallotProcess::workermaxport (C++ <i>function</i> ), 163, 727, 1334
sh::base::ShallotProcess::configurationValueBaseAssignsh (C++ <i>function</i> ), 163, 727, 1334	sh::base::ShallotProcess::workerminport (C++ <i>function</i> ), 163, 727, 1334
sh::base::ShallotProcess::configurationValueFiles (C++ <i>function</i> ), 163, 727, 1334	sh::base::Singleton (C++ <i>class</i> ), 165, 729, 1336
sh::base::ShallotProcess::doInitialize (C++ <i>function</i> ), 163, 728, 1335	sh::base::Singleton::_mutex_shutdown (C++ <i>member</i> ), 165, 730, 1337
sh::base::ShallotProcess::doShutdown (C++ <i>function</i> ), 164, 728, 1335	sh::base::Singleton::_shutdown (C++ <i>member</i> ), 165, 730, 1337
sh::base::ShallotProcess::homepage (C++ <i>function</i> ), 164, 728, 1335	sh::base::Singleton::~Singleton (C++ <i>function</i> ), 165, 730, 1337
sh::base::ShallotProcess::initialWorkDirectory (C++ <i>function</i> ), 163, 727, 1334	sh::base::Singleton::doInitialize (C++ <i>function</i> ), 165, 730, 1337
sh::base::ShallotProcess::isAlive (C++ <i>function</i> ), 164, 728, 1335	sh::base::Singleton::doShutdown (C++ <i>function</i> ), 165, 730, 1337
sh::base::ShallotProcess::lang (C++ <i>function</i> ), 163, 727, 1334	sh::base::Singleton::isAlive (C++ <i>function</i> ), 165, 730, 1337



tion), 165, 730, 1337

sh::base::Singleton::shutdown (C++ function), 165, 730, 1337

sh::base::Singleton::Singleton (C++ function), 165, 730, 1337

sh::base::SingletonInitializer (C++ class), 166, 730, 1337

sh::base::SingletonInitializer::\_groups (C++ member), 167, 731, 1338

sh::base::SingletonInitializer::\_isshutdown (C++ member), 167, 731, 1338

sh::base::SingletonInitializer::\_singleton (C++ member), 167, 731, 1338

sh::base::SingletonInitializer::~SingletonInitializer (C++ function), 166, 731, 1337

sh::base::SingletonInitializer::callInitShlib (C++ function), 166, 731, 1337

sh::base::SingletonInitializer::dependsOn (C++ function), 167, 731, 1338

sh::base::SingletonInitializer::initializeThread (C++ function), 166, 731, 1338

sh::base::SingletonInitializer::isShutdown (C++ function), 166, 731, 1338

sh::base::SingletonInitializer::registerGroup (C++ function), 166, 731, 1338

sh::base::SingletonInitializer::registerSingleton (C++ function), 166, 731, 1338

sh::base::SingletonInitializer::shutdown (C++ function), 166, 731, 1337

sh::base::SingletonInitializer::shutdownState (C++ member), 167, 731, 1338

sh::base::SingletonInitializer::SingletonGroupType (C++ type), 167, 731, 1338

sh::base::SingletonInitializer::SingletonInitializer (C++ function), 166, 731, 1337

sh::base::ThreadDispatcher (C++ class), 167, 731, 1338

sh::base::ThreadDispatcher::\_invokeAsync (C++ function), 168, 732, 1339

sh::base::ThreadDispatcher::\_invoked (C++ function), 168, 732, 1339

sh::base::ThreadDispatcher::\_queue (C++ member), 168, 732, 1339

sh::base::ThreadDispatcher::\_thread (C++ member), 168, 732, 1339

sh::base::ThreadDispatcher::callsmutex (C++ member), 168, 732, 1339

sh::base::ThreadDispatcher::inThread (C++ function), 167, 732, 1339

sh::base::ThreadDispatcher::invokeAsync (C++ function), 167, 732, 1339

sh::base::ThreadDispatcher::invokeSync (C++ function), 167, 732, 1339

sh::base::ThreadDispatcher::slot\_invoked (C++ function), 168, 732, 1339

sh::base::ThreadDispatcher::ThreadDispatcher (C++ function), 167, 732, 1339

sh::base::ThreadPool (C++ class), 168, 732, 1339

sh::base::ThreadPool::\_enqueueForce (C++ function), 169, 733, 1340

sh::base::ThreadPool::\_shutdown (C++ member), 169, 733, 1340

sh::base::ThreadPool::\_threads (C++ member), 169, 733, 1340

sh::base::ThreadPool::\_threadsalive (C++ member), 169, 733, 1340

sh::base::ThreadPool::doInitialize (C++ function), 169, 733, 1340

sh::base::ThreadPool::doShutdown (C++ function), 168, 733, 1340

sh::base::ThreadPool::enqueueForce (C++ function), 168, 733, 1340

sh::base::ThreadPool::enqueueForceBeyondAsyncCallB (C++ function), 168, 733, 1340

sh::base::ThreadPool::enqueueIfIn (C++ function), 168, 733, 1340

sh::base::ThreadPool::enqueueIfInMain (C++ function), 168, 733, 1340

sh::base::ThreadPool::getThreadCount (C++ function), 169, 733, 1340

sh::base::ThreadPool::isShutdown (C++ function), 168, 733, 1340

sh::base::ThreadPool::isShuttingDown (C++ function), 168, 733, 1340

sh::base::ThreadPool::pooladdedcondition (C++ member), 169, 733, 1340

sh::base::ThreadPool::poolmutex (C++ member), 169, 733, 1340

sh::base::ThreadPool::respectThreadAbort (C++ function), 169, 733, 1340

sh::base::ThreadPool::tasks (C++ member), 169, 733, 1340

sh::base::ThreadPool::THREAD\_COUNT (C++ member), 169, 733, 1340

sh::base::ThreadPoolThread (C++ class), 169, 734, 1341

sh::configuration (C++ type), 734, 1341

sh::configuration::ConfigurationCategory (C++ enum), 734, 1341

sh::configuration::ConfigurationCategory::CategoryY (C++ enumerator), 734, 1341

sh::configuration::ConfigurationCategory::CategoryY (C++ enumerator), 734, 1341

sh::configuration::ConfigurationCategory::CategoryY (C++ enumerator), 734, 1341

sh::configuration::ConfigurationCategory::CategoryY (C++ enumerator), 734, 1341

sh::configuration::ConfigurationCategory::CategoryY (C++ enumerator), 734, 1341

sh::configuration::ConfigurationManager sh::configuration::ConfigurationManager::Configurat  
 (C++ class), 169, 734, 1341 (C++ function), 171, 173, 736, 1343  
 sh::configuration::ConfigurationManager:sh::configuration::ConfigurationManager::Configurat  
 (C++ member), 170, 735, 1342 (C++ function), 171, 173, 736, 1343  
 sh::configuration::ConfigurationManager:sh::fixedconfigvalue::ConfigurationManager::doInitial  
 (C++ member), 170, 735, 1342 (C++ function), 170, 734, 1342  
 sh::configuration::ConfigurationManager:sh::fixedconfigvalue::ConfiguraitizedManager::doShutdown  
 (C++ member), 170, 735, 1342 (C++ function), 170, 735, 1342  
 sh::configuration::ConfigurationManager:sh::mutexconfiguration::ConfigurationManager::getAllCon  
 (C++ member), 170, 735, 1342 (C++ function), 170, 734, 1341  
 sh::configuration::ConfigurationManager:sh::configurationManager::ConfigurationManager::getFixedC  
 (C++ function), 170, 735, 1342 (C++ function), 170, 735, 1342  
 sh::configuration::ConfigurationManager:sh::configurationValue::ConfigurationManager::isAlive  
 (C++ class), 170, 172, 735, 1342 (C++ function), 170, 735, 1342  
 sh::configuration::ConfigurationManager:sh::configurationValue::ConfigurationManager::registerC  
 (C++ function), 171, 172, 736, 1343 (C++ function), 170, 734, 1341  
 sh::configuration::ConfigurationManager:sh::configurationValue::ConfigurationManager::shutdown  
 (C++ function), 171, 172, 736, 1343 (C++ function), 170, 735, 1342  
 sh::configuration::ConfigurationManager:sh::configurationValue::ConfigurationManager::ValueImpl  
 (C++ function), 170, 172, 735, 1342 (C++ class), 173, 736, 1343  
 sh::configuration::ConfigurationManager:sh::configurationValue::ConfigurationManager::category  
 (C++ function), 171, 172, 735, 1342 (C++ function), 174, 737, 1344  
 sh::configuration::ConfigurationManager:sh::configurationValue::ConfigurationManager::changeHint  
 (C++ function), 171, 172, 736, 1343 (C++ function), 174, 737, 1344  
 sh::configuration::ConfigurationManager:sh::configurationValue::ConfigurationManager::Configuratio  
 (C++ function), 171, 172, 736, 1343 (C++ function), 173, 737, 1344  
 sh::configuration::ConfigurationManager:sh::configurationValue::ConfigurationManager::BonsumeValue  
 (C++ function), 171, 172, 735, 1342 (C++ function), 174, 737, 1344  
 sh::configuration::ConfigurationManager:sh::configurationValue::ConfigurationManager::DefaultValue  
 (C++ function), 171, 172, 735, 1342 (C++ function), 174, 737, 1344  
 sh::configuration::ConfigurationManager:sh::configurationValue::ConfigurationManager::description  
 (C++ function), 171, 172, 735, 1342 (C++ function), 174, 737, 1344  
 sh::configuration::ConfigurationManager:sh::configurationValue::ConfigurationManager::getConfV  
 (C++ function), 171, 172, 735, 1342 (C++ function), 173, 737, 1344  
 sh::configuration::ConfigurationManager:sh::configurationValue::ConfigurationManager::getConfV  
 (C++ function), 170, 172, 735, 1342 (C++ function), 173, 737, 1344  
 sh::configuration::ConfigurationManager:sh::configurationValue::ConfigurationManager::getConfV  
 (C++ function), 171, 172, 736, 1343 (C++ function), 173, 737, 1344  
 sh::configuration::ConfigurationManager:sh::configurationValue::ConfigurationManager::getConfV  
 (C++ function), 170, 172, 735, 1342 (C++ function), 173, 737, 1344  
 sh::configuration::ConfigurationManager:sh::configurationValue::ConfigurationManager::getConfV  
 (C++ function), 171, 172, 736, 1343 (C++ function), 173, 737, 1344  
 sh::configuration::ConfigurationManager:sh::configurationValue::ConfigurationManager::getConfV  
 (C++ member), 172, 173, 736, 1343 (C++ function), 173, 737, 1344  
 sh::configuration::ConfigurationManager:sh::configurationValue::ConfigurationManager::longDescript  
 (C++ function), 171, 172, 736, 1343 (C++ function), 174, 737, 1344  
 sh::configuration::ConfigurationManager:sh::configurationValue::ConfigurationManager::mayChange  
 (C++ function), 170, 172, 735, 1342 (C++ function), 173, 737, 1344  
 sh::configuration::ConfigurationManager:sh::configurationValue::ConfigurationManager::name  
 (C++ function), 171, 172, 736, 1343 (C++ function), 174, 737, 1344  
 sh::configuration::ConfigurationManager:sh::configurationValue::ConfigurationManager::BooleanConfigVa  
 (C++ function), 171, 173, 736, 1343 (C++ function), 173, 737, 1344  
 sh::configuration::ConfigurationManager:sh::configurationValue::ConfigurationManager::FloatvalueType  
 (C++ function), 171, 173, 736, 1343 (C++ function), 174, 737, 1344  
 sh::configuration::ConfigurationManager:sh::configurationValue::ConfigurationManager::IntegerValueTypeBoo  
 (C++ function), 171, 173, 736, 1343 (C++ function), 174, 738, 1345





(C++ function), 180, 743, 1350  
 sh::detailcolumns::DetailColumnDirectSymlinkTarget (C++ member), 181, 745, 1352  
 (C++ function), 179, 742, 1349  
 sh::detailcolumns::DetailColumnDirectSymlinkTarget (C++ member), 181, 745, 1352  
 (C++ member), 180, 743, 1350  
 sh::detailcolumns::DetailColumnDirectSymlinkTarget (C++ member), 181, 745, 1352  
 (C++ member), 180, 743, 1350  
 sh::detailcolumns::DetailColumnDirectSymlinkTarget (C++ function), 180, 743, 1350  
 (C++ member), 180, 743, 1350  
 sh::detailcolumns::DetailColumnDirectSymlinkTarget (C++ function), 180, 743, 1350  
 (C++ function), 178, 741, 1348  
 sh::detailcolumns::DetailColumnDirectSymlinkTarget (C++ function), 181, 744, 1351  
 (C++ function), 178, 741, 1348  
 sh::detailcolumns::DetailColumnDirectSymlinkTarget (C++ function), 181, 744, 1351  
 (C++ function), 179, 742, 1349  
 sh::detailcolumns::DetailColumnDirectSymlinkTarget (C++ function), 181, 744, 1351  
 (C++ function), 179, 742, 1350  
 sh::detailcolumns::DetailColumnDirectSymlinkTarget (C++ function), 181, 744, 1351  
 (C++ function), 179, 742, 1349  
 sh::detailcolumns::DetailColumnDirectSymlinkTarget (C++ function), 181, 744, 1351  
 (C++ function), 179, 742, 1349  
 sh::detailcolumns::DetailColumnDirectSymlinkTarget (C++ function), 181, 744, 1351  
 (C++ function), 178, 741, 1348  
 sh::detailcolumns::DetailColumnDirectSymlinkTarget (C++ function), 180, 743, 1350  
 (C++ function), 178, 741, 1348  
 sh::detailcolumns::DetailColumnDirectSymlinkTarget (C++ function), 181, 744, 1351  
 (C++ function), 179, 742, 1350  
 sh::detailcolumns::DetailColumnDirectSymlinkTarget (C++ function), 180, 743, 1350  
 (C++ function), 178, 741, 1348  
 sh::detailcolumns::DetailColumnDirectSymlinkTarget (C++ function), 181, 744, 1351  
 (C++ function), 179, 742, 1349  
 sh::detailcolumns::DetailColumnDirectSymlinkTarget (C++ function), 180, 743, 1350  
 (C++ function), 178, 741, 1348  
 sh::detailcolumns::DetailColumnDirectSymlinkTarget (C++ function), 181, 744, 1351  
 (C++ function), 179, 742, 1349  
 sh::detailcolumns::DetailColumnDirectSymlinkTarget (C++ function), 180, 743, 1350  
 (C++ function), 178, 741, 1348  
 sh::detailcolumns::DetailColumnDirectSymlinkTarget (C++ function), 181, 744, 1351  
 (C++ function), 179, 742, 1349  
 sh::detailcolumns::DetailColumnDirectSymlinkTarget (C++ function), 180, 743, 1350  
 (C++ function), 178, 741, 1348  
 sh::detailcolumns::DetailColumnDirectSymlinkTarget (C++ function), 181, 744, 1351  
 (C++ function), 179, 742, 1350  
 sh::detailcolumns::DetailColumnExtendedAttributes (C++ class), 182, 745, 1352  
 (C++ class), 180, 743, 1350  
 sh::detailcolumns::DetailColumnExtendedAttributes (C++ function), 182, 745, 1352  
 (C++ function), 180, 181, 743, 744, 1350, 1351  
 sh::detailcolumns::DetailColumnExtendedAttributes (C++ function), 183, 746, 1353  
 sh::detailcolumns::DetailColumnExtendedAttributes::applyValue (C++ function), 182, 745, 1352  
 sh::detailcolumns::DetailColumnExtendedAttributes::computeValue (C++ function), 182, 745, 1352  
 sh::detailcolumns::DetailColumnExtendedAttributes::defaultWidth (C++ function), 182, 745, 1352  
 sh::detailcolumns::DetailColumnExtendedAttributes::defaultWidth (C++ function), 183, 746, 1353  
 sh::detailcolumns::DetailColumnExtendedAttributes::displayIndex (C++ function), 182, 745, 1352  
 sh::detailcolumns::DetailColumnExtendedAttributes::DISPLAYINDEX (C++ member), 183, 746, 1353

sh::detailcolumns::DetailColumnFileSize:sh::detailcolumns::DetailColumnMimetype::displayNa  
 (C++ member), 183, 746, 1353 (C++ function), 184, 747, 1354  
 sh::detailcolumns::DetailColumnFileSize:sh::detailcolumns::DetailColumnMimetype::doShutdw  
 (C++ member), 183, 746, 1353 (C++ function), 184, 748, 1354  
 sh::detailcolumns::DetailColumnFileSize:sh::detailcolumns::DetailColumnMimetype::findKnown  
 (C++ function), 182, 745, 1352 (C++ function), 185, 748, 1355  
 sh::detailcolumns::DetailColumnFileSize:sh::detailcolumns::DetailColumnMimetype::isAlive  
 (C++ function), 183, 746, 1353 (C++ function), 185, 748, 1355  
 sh::detailcolumns::DetailColumnFileSize:sh::detailcolumns::DetailColumnMimetype::isRightAl  
 (C++ function), 183, 746, 1353 (C++ function), 184, 747, 1354  
 sh::detailcolumns::DetailColumnFileSize:sh::detailcolumns::DetailColumnMimetype::isValueAv  
 (C++ function), 183, 746, 1353 (C++ function), 184, 747, 1354  
 sh::detailcolumns::DetailColumnFileSize:sh::detailcolumns::DetailColumnMimetype::isVisible  
 (C++ function), 182, 745, 1352 (C++ function), 184, 747, 1354  
 sh::detailcolumns::DetailColumnFileSize:sh::detailcolumns::DetailColumnMimetype::name  
 (C++ function), 182, 745, 1352 (C++ function), 184, 747, 1354  
 sh::detailcolumns::DetailColumnFileSize:sh::detailcolumns::DetailColumnMimetype::registerK  
 (C++ function), 182, 745, 1352 (C++ function), 185, 748, 1355  
 sh::detailcolumns::DetailColumnFileSize:sh::detailcolumns::DetailColumnMimetype::requestVa  
 (C++ function), 182, 745, 1352 (C++ function), 184, 747, 1354  
 sh::detailcolumns::DetailColumnFileSize:sh::detailcolumns::DetailColumnMimetype::shutdown  
 (C++ function), 183, 746, 1353 (C++ function), 185, 748, 1355  
 sh::detailcolumns::DetailColumnFileSize:sh::detailcolumns::DetailColumnMimetype::sort\_doTyp  
 (C++ function), 182, 745, 1352 (C++ function), 184, 747, 1354  
 sh::detailcolumns::DetailColumnFileSize:sh::detailcolumns::DetailColumnMimetype::sort\_order  
 (C++ function), 183, 746, 1353 (C++ function), 184, 747, 1354  
 sh::detailcolumns::DetailColumnFileSize:sh::detailcolumns::DetailColumnMimetype::value  
 (C++ function), 182, 745, 1352 (C++ function), 184, 747, 1354  
 sh::detailcolumns::DetailColumnFileSize:sh::detailcolumns::DetailColumnMimetype::VALUE\_UNAV  
 (C++ function), 182, 745, 1352 (C++ function), 185, 748, 1355  
 sh::detailcolumns::DetailColumnFileSize:sh::detailcolumns::DetailColumnMtime  
 (C++ function), 182, 745, 1352 (C++ class), 185, 748, 1355  
 sh::detailcolumns::DetailColumnFileSize:sh::detailcolumns::DetailColumnMtime::applyValueByL  
 (C++ function), 183, 746, 1353 (C++ function), 185, 186, 748, 749, 1355,  
 sh::detailcolumns::DetailColumnMimetype 1356  
 (C++ class), 183, 746, 1353 sh::detailcolumns::DetailColumnMtime::applyValueByL  
 sh::detailcolumns::DetailColumnMimetype::applyValueByL (C++ function), 186, 749, 1356  
 (C++ function), 184, 747, 1354 sh::detailcolumns::DetailColumnMtime::computeValue  
 sh::detailcolumns::DetailColumnMimetype::applyValueByL (C++ function), 186, 749, 1356  
 (C++ function), 184, 747, 1354 sh::detailcolumns::DetailColumnMtime::defaultWidth  
 sh::detailcolumns::DetailColumnMimetype::computeValue (C++ function), 186, 749, 1356  
 (C++ function), 184, 747, 1354 sh::detailcolumns::DetailColumnMtime::DetailColumnM  
 sh::detailcolumns::DetailColumnMimetype::defaultWidth (C++ function), 187, 750, 1357  
 (C++ function), 184, 747, 1354 sh::detailcolumns::DetailColumnMtime::displayIndex  
 sh::detailcolumns::DetailColumnMimetype::DetailColumnMtime (C++ function), 186, 749, 1356  
 (C++ function), 185, 748, 1355 sh::detailcolumns::DetailColumnMtime::DISPLAYINDEX  
 sh::detailcolumns::DetailColumnMimetype::displayIndex (C++ member), 187, 750, 1357  
 (C++ function), 184, 747, 1354 sh::detailcolumns::DetailColumnMtime::DISPLAYINDEX  
 sh::detailcolumns::DetailColumnMimetype::DISPLAYINDEX (C++ member), 187, 750, 1357  
 (C++ member), 185, 748, 1355 sh::detailcolumns::DetailColumnMtime::DISPLAYINDEX  
 sh::detailcolumns::DetailColumnMimetype::DISPLAYINDEX (C++ member), 187, 750, 1357  
 (C++ member), 185, 748, 1355 sh::detailcolumns::DetailColumnMtime::displayName  
 sh::detailcolumns::DetailColumnMimetype::DISPLAYINDEX (C++ function), 185, 748, 1355  
 (C++ member), 185, 748, 1355 sh::detailcolumns::DetailColumnMtime::doShutdown



sh::exceptions::ArgumentException::isProgramException (C++ function), 189, 752, 1359  
 sh::exceptions::ArgumentException::isResumeable (C++ function), 189, 753, 1359  
 sh::exceptions::ArgumentException::isRetryable (C++ function), 189, 753, 1359  
 sh::exceptions::ArgumentException::isRuntimeException (C++ function), 189, 752, 1359  
 sh::exceptions::ArgumentException::message (C++ function), 189, 752, 1359  
 sh::exceptions::ArgumentException::name (C++ function), 189, 752, 1359  
 sh::exceptions::ArgumentException::setCustomValue (C++ function), 189, 753, 1359  
 sh::exceptions::ArgumentException::setResumeable (C++ function), 189, 753, 1359  
 sh::exceptions::ArgumentException::setRetryable (C++ function), 189, 753, 1359  
 sh::exceptions::ArgumentException::SHALLOT\_MUST\_CLOSE\_TEXT (C++ member), 190, 753, 1360  
 sh::exceptions::ArgumentException::UNDEFINED\_ERROR\_OCCURRED (C++ member), 190, 753, 1360  
 sh::exceptions::ArgumentException::Values (C++ member), 190, 753, 1360  
 sh::exceptions::CancelException (C++ class), 190, 753, 1360  
 sh::exceptions::CancelException::CancelException (C++ function), 190, 754, 1360  
 sh::exceptions::Exception (C++ class), 190, 754, 1360  
 sh::exceptions::Exception::\_demangle (C++ function), 191, 755, 1361  
 sh::exceptions::Exception::\_inShutdown (C++ member), 192, 755, 1362  
 sh::exceptions::Exception::\_inShutdownMutex (C++ member), 192, 755, 1362  
 sh::exceptions::Exception::autoRetryRecommended (C++ function), 191, 754, 1361  
 sh::exceptions::Exception::auxiliary (C++ function), 191, 754, 1360  
 sh::exceptions::Exception::callstack (C++ function), 191, 754, 1360  
 sh::exceptions::Exception::classes (C++ function), 191, 754, 1360  
 sh::exceptions::Exception::createRegistersHandler (C++ function), 191, 754, 1361  
 sh::exceptions::Exception::customValue (C++ function), 191, 754, 1360  
 sh::exceptions::Exception::data (C++ function), 191, 754, 1361  
 sh::exceptions::Exception::details (C++ function), 191, 754, 1360  
 sh::exceptions::Exception::Exception (C++ function), 191, 754, 1361  
 sh::exceptions::Exception::exceptionDialog (C++ function), 191, 192, 755, 1361, 1362  
 sh::exceptions::Exception::executeGuarded (C++ function), 191, 754, 1361  
 sh::exceptions::Exception::executeGuarded\_errorpan (C++ function), 191, 755, 1361  
 sh::exceptions::Exception::HandlerSettings (C++ class), 192, 194, 755, 1362  
 sh::exceptions::Exception::HandlerSettings::cancel (C++ member), 192, 194, 755, 1362  
 sh::exceptions::Exception::HandlerSettings::except (C++ member), 192, 194, 755, 1362  
 sh::exceptions::Exception::HandlerSettings::except (C++ member), 192, 194, 755, 1362  
 sh::exceptions::Exception::HandlerSettings::except (C++ member), 192, 194, 755, 1362  
 sh::exceptions::Exception::HandlerSettings::except (C++ member), 192, 194, 755, 1362  
 sh::exceptions::Exception::HandlerSettings::Handler (C++ function), 192, 194, 755, 1362  
 sh::exceptions::Exception::isClass (C++ function), 191, 754, 1361  
 sh::exceptions::Exception::isDetailsAreInteresting (C++ function), 191, 754, 1361  
 sh::exceptions::Exception::isProgramException (C++ function), 191, 754, 1361  
 sh::exceptions::Exception::isResumeable (C++ function), 191, 754, 1361  
 sh::exceptions::Exception::isRetryable (C++ function), 191, 754, 1361  
 sh::exceptions::Exception::isRuntimeException (C++ function), 191, 754, 1360  
 sh::exceptions::Exception::message (C++ function), 191, 754, 1360  
 sh::exceptions::Exception::name (C++ function), 191, 754, 1360  
 sh::exceptions::Exception::RegisterHandler (C++ class), 192, 195, 756, 1362  
 sh::exceptions::Exception::RegisterHandler::\_stack (C++ member), 193, 195, 756, 1363  
 sh::exceptions::Exception::RegisterHandler::~Register (C++ function), 193, 195, 756, 1362  
 sh::exceptions::Exception::RegisterHandler::Register (C++ function), 193, 195, 756, 1362  
 sh::exceptions::Exception::setCustomValue (C++ function), 191, 754, 1361  
 sh::exceptions::Exception::setResumeable (C++ function), 191, 754, 1361  
 sh::exceptions::Exception::setReTryable (C++ function), 191, 754, 1361  
 sh::exceptions::Exception::SHALLOT\_MUST\_CLOSE\_TEXT (C++ member), 192, 755, 1361  
 sh::exceptions::Exception::UNDEFINED\_ERROR\_OCCURRED (C++ member), 192, 755, 1361





sh::exceptions::IOException::setCustomValue (C++ function), 195, 758, 1364  
 sh::exceptions::IOException::setResumeable (C++ function), 195, 758, 1364  
 sh::exceptions::IOException::setRetryable (C++ function), 195, 758, 1364  
 sh::exceptions::IOException::SHALLOT\_MUST\_THROW (C++ member), 196, 758, 1365  
 sh::exceptions::IOException::UNDEFINED\_ERROR (C++ member), 196, 758, 1365  
 sh::exceptions::IOException::Value\_isShallot (C++ member), 196, 758, 1365  
 sh::exceptions::PermissionDeniedException (C++ class), 196, 758, 1365  
 sh::exceptions::PermissionDeniedException::\_demangle (C++ function), 197, 759, 1366  
 sh::exceptions::PermissionDeniedException::auxiliary (C++ function), 197, 759, 1366  
 sh::exceptions::PermissionDeniedException::callstack (C++ function), 197, 759, 1366  
 sh::exceptions::PermissionDeniedException::classes (C++ function), 197, 759, 1366  
 sh::exceptions::PermissionDeniedException::createRegisterHandler (C++ function), 197, 759, 1366  
 sh::exceptions::PermissionDeniedException::customValue (C++ function), 197, 759, 1366  
 sh::exceptions::PermissionDeniedException::data (C++ function), 197, 759, 1366  
 sh::exceptions::PermissionDeniedException::details (C++ function), 197, 759, 1366  
 sh::exceptions::PermissionDeniedException::exceptionDialog (C++ function), 197, 759, 1366  
 sh::exceptions::PermissionDeniedException::executeGuarded (C++ function), 197, 759, 1366  
 sh::exceptions::PermissionDeniedException::executeGuarded\_exception (C++ function), 197, 759, 1366  
 sh::exceptions::PermissionDeniedException::isClass (C++ function), 197, 759, 1365  
 sh::exceptions::PermissionDeniedException::isDetailsAreInternal (C++ function), 197, 759, 1365  
 sh::exceptions::PermissionDeniedException::isProgramException (C++ function), 197, 759, 1365  
 sh::exceptions::PermissionDeniedException::isResumeable (C++ function), 197, 759, 1365  
 sh::exceptions::PermissionDeniedException::isRetryable (C++ function), 197, 759, 1365  
 sh::exceptions::PermissionDeniedException::isRuntimeException (C++ function), 196, 758, 1365  
 sh::exceptions::PermissionDeniedException::message (C++ function), 196, 758, 1365  
 sh::exceptions::PermissionDeniedException::Permissions (C++ function), 196, 758, 1365  
 sh::exceptions::PermissionDeniedException::setCustomValue (C++ function), 197, 759, 1366  
 sh::exceptions::PermissionDeniedException::setResumeable (C++ function), 197, 759, 1366  
 sh::exceptions::PermissionDeniedException::setRetryable (C++ function), 197, 759, 1366  
 sh::exceptions::PermissionDeniedException::SHALLOT\_MUST\_THROW (C++ member), 197, 759, 1365  
 sh::exceptions::PermissionDeniedException::UNDEFINED\_ERROR (C++ member), 197, 759, 1366  
 sh::exceptions::PermissionDeniedException::Value\_isShallot (C++ member), 197, 759, 1366  
 sh::exceptions::ProgramException (C++ class), 198, 759, 1366  
 sh::exceptions::ProgramException::\_demangle (C++ function), 199, 760, 1367  
 sh::exceptions::ProgramException::auxiliary (C++ function), 198, 760, 1367  
 sh::exceptions::ProgramException::callstack (C++ function), 198, 760, 1366  
 sh::exceptions::ProgramException::classes (C++ function), 198, 760, 1366  
 sh::exceptions::ProgramException::createRegisterHandler (C++ function), 198, 760, 1367  
 sh::exceptions::ProgramException::customValue (C++ function), 198, 760, 1366  
 sh::exceptions::ProgramException::data (C++ function), 198, 760, 1367  
 sh::exceptions::ProgramException::details (C++ function), 198, 760, 1366  
 sh::exceptions::ProgramException::exceptionDialog (C++ function), 199, 760, 1367  
 sh::exceptions::ProgramException::executeGuarded (C++ function), 198, 760, 1367  
 sh::exceptions::ProgramException::executeGuarded\_exception (C++ function), 198, 760, 1367  
 sh::exceptions::ProgramException::isClass (C++ function), 198, 760, 1367  
 sh::exceptions::ProgramException::isDetailsAreInternal (C++ function), 198, 760, 1367  
 sh::exceptions::ProgramException::isProgramException (C++ function), 198, 760, 1366  
 sh::exceptions::ProgramException::isResumeable (C++ function), 198, 760, 1367  
 sh::exceptions::ProgramException::isRetryable (C++ function), 198, 760, 1367  
 sh::exceptions::ProgramException::isRuntimeException (C++ function), 198, 760, 1366  
 sh::exceptions::ProgramException::message (C++ function), 198, 760, 1366

sh::exceptions::ProgramException::name (C++ function), 198, 760, 1366  
 sh::exceptions::ProgramException::ProgramException (C++ function), 198, 760, 1366  
 sh::exceptions::ProgramException::setCustomValue (C++ function), 198, 760, 1367  
 sh::exceptions::ProgramException::setResumeable (C++ function), 198, 760, 1367  
 sh::exceptions::ProgramException::setRetryable (C++ function), 198, 760, 1367  
 sh::exceptions::ProgramException::SHALLOT\_MUST\_CLOSE (C++ member), 199, 761, 1367  
 sh::exceptions::ProgramException::UNDEFINED\_ERROR\_OCCURRED (C++ member), 199, 761, 1367  
 sh::exceptions::ProgramException::Value\_isShallotException (C++ member), 199, 761, 1367  
 sh::exceptions::RuntimeException (C++ class), 199, 761, 1367  
 sh::exceptions::RuntimeException::\_demangle (C++ function), 200, 762, 1368  
 sh::exceptions::RuntimeException::autoRetryRecommendedInThreadAbortException (C++ function), 199, 761, 1368  
 sh::exceptions::RuntimeException::auxiliary (C++ function), 199, 761, 1368  
 sh::exceptions::RuntimeException::callstack (C++ function), 199, 761, 1368  
 sh::exceptions::RuntimeException::classes (C++ function), 199, 761, 1368  
 sh::exceptions::RuntimeException::createRegisterHandler (C++ function), 200, 762, 1368  
 sh::exceptions::RuntimeException::customValue (C++ function), 199, 761, 1368  
 sh::exceptions::RuntimeException::data (C++ function), 199, 761, 1368  
 sh::exceptions::RuntimeException::detail (C++ function), 199, 761, 1368  
 sh::exceptions::RuntimeException::exceptionDialog (C++ function), 200, 762, 1368  
 sh::exceptions::RuntimeException::executeGuarded (C++ function), 200, 762, 1368  
 sh::exceptions::RuntimeException::executeGuardedPersons (C++ function), 200, 762, 1368  
 sh::exceptions::RuntimeException::isClass (C++ function), 199, 761, 1368  
 sh::exceptions::RuntimeException::isDetailsAreInteresting (C++ function), 199, 761, 1368  
 sh::exceptions::RuntimeException::isProgramException (C++ function), 199, 761, 1368  
 sh::exceptions::RuntimeException::isResumeable (C++ function), 199, 761, 1368  
 sh::exceptions::RuntimeException::isRetryable (C++ function), 199, 761, 1368  
 sh::exceptions::RuntimeException::isRuntimeException (C++ function), 199, 761, 1368  
 sh::exceptions::RuntimeException::message (C++ function), 199, 761, 1368  
 sh::exceptions::RuntimeException::name (C++ function), 199, 761, 1368  
 sh::exceptions::RuntimeException::RuntimeException (C++ function), 199, 761, 1368  
 sh::exceptions::RuntimeException::RuntimeException::RuntimeException (C++ function), 199, 761, 1368  
 sh::exceptions::RuntimeException::setCustomValue (C++ function), 199, 761, 1368  
 sh::exceptions::RuntimeException::setResumeable (C++ function), 199, 761, 1368  
 sh::exceptions::RuntimeException::setRetryable (C++ function), 199, 761, 1368  
 sh::exceptions::RuntimeException::SHALLOT\_MUST\_CLOSE (C++ member), 200, 762, 1369  
 sh::exceptions::RuntimeException::UNDEFINED\_ERROR\_OCCURRED (C++ member), 200, 762, 1369  
 sh::exceptions::RuntimeException::Value\_isShallotException (C++ member), 200, 762, 1369  
 sh::exceptions::ThreadAbortException (C++ class), 200, 762, 1369  
 sh::exceptions::ThreadAbortException::ThreadAbortException (C++ function), 200, 762, 1369  
 sh::exceptions::ThreadingException (C++ class), 200, 762, 1369  
 sh::exceptions::ThreadingException::\_demangle (C++ function), 201, 763, 1370  
 sh::exceptions::ThreadingException::autoRetryRecommendedInThreadAbortException (C++ function), 201, 763, 1369  
 sh::exceptions::ThreadingException::auxiliary (C++ function), 201, 762, 1369  
 sh::exceptions::ThreadingException::callstack (C++ function), 201, 762, 1369  
 sh::exceptions::ThreadingException::classes (C++ function), 201, 762, 1369  
 sh::exceptions::ThreadingException::createRegisterHandler (C++ function), 201, 763, 1370  
 sh::exceptions::ThreadingException::customValue (C++ function), 201, 762, 1369  
 sh::exceptions::ThreadingException::data (C++ function), 201, 763, 1369  
 sh::exceptions::ThreadingException::details (C++ function), 201, 762, 1369  
 sh::exceptions::ThreadingException::exceptionDialog (C++ function), 201, 763, 1370  
 sh::exceptions::ThreadingException::executeGuarded (C++ function), 201, 763, 1370  
 sh::exceptions::ThreadingException::executeGuardedPersons (C++ function), 201, 763, 1370  
 sh::exceptions::ThreadingException::isClass (C++ function), 201, 763, 1369  
 sh::exceptions::ThreadingException::isDetailsAreInteresting (C++ function), 201, 763, 1370  
 sh::exceptions::ThreadingException::isProgramException (C++ function), 201, 763, 1370  
 sh::exceptions::ThreadingException::isResumeable (C++ function), 201, 763, 1369  
 sh::exceptions::ThreadingException::isRetryable (C++ function), 201, 763, 1369  
 sh::exceptions::ThreadingException::isRuntimeException (C++ function), 201, 762, 1369

sh::exceptions::ThreadingException::isReshmeablefilepropertydialogtabs::FilePropertyDialogTabEx  
(C++ function), 201, 763, 1369 (C++ function), 204

sh::exceptions::ThreadingException::isReshyafilepropertydialogtabs::FilePropertyDialogTabEx  
(C++ function), 201, 763, 1369 (C++ function), 204

sh::exceptions::ThreadingException::isRushimefilepropertydialogtabs::FilePropertyDialogTabEx  
(C++ function), 201, 762, 1369 (C++ function), 204

sh::exceptions::ThreadingException::message:filepropertydialogtabs::FilePropertyDialogTabEx  
(C++ function), 201, 762, 1369 (C++ function), 203

sh::exceptions::ThreadingException::namesh::filepropertydialogtabs::FilePropertyDialogTabEx  
(C++ function), 201, 762, 1369 (C++ function), 203

sh::exceptions::ThreadingException::setCstionfilepropertydialogtabs::FilePropertyDialogTabEx  
(C++ function), 201, 763, 1369 (C++ function), 203

sh::exceptions::ThreadingException::setReshmeablefilepropertydialogtabs::FilePropertyDialogTabEx  
(C++ function), 201, 763, 1369 (C++ function), 203

sh::exceptions::ThreadingException::setResTryfilepropertydialogtabs::FilePropertyDialogTabEx  
(C++ function), 201, 763, 1369 (C++ function), 204

sh::exceptions::ThreadingException::SHALLOT\_MUTEX\_CLOSE\_FILE dialogtabs::FilePropertyDialogTabEx  
(C++ member), 202, 763, 1370 (C++ function), 203

sh::exceptions::ThreadingException::ThreadingExceptionfilepropertydialogtabs::FilePropertyDialogTabEx  
(C++ function), 201, 762, 1369 (C++ function), 203

sh::exceptions::ThreadingException::UNDEFINED\_ERROR\_OCCURRED dialogtabs::FilePropertyDialogTabEx  
(C++ member), 202, 763, 1370 (C++ function), 203

sh::exceptions::ThreadingException::ValuehisShallopException dialogtabs::FilePropertyDialogTabEx  
(C++ member), 202, 763, 1370 (C++ function), 204

sh::filepropertydialogtabs (C++ type), 763, sh::filepropertydialogtabs::FilePropertyDialogTabEx  
1370 (C++ class), 204

sh::filepropertydialogtabs::FilePropertyDialogTabExpendyDialogbabe:FilePropertyDialogTabEx  
(C++ class), 202, 763, 1370 (C++ function), 205

sh::filepropertydialogtabs::FilePropertyDialogTabExpendyDialogbabe:FactePonAddAyDiabogTabEx  
(C++ class), 203 (C++ function), 204

sh::filepropertydialogtabs::FilePropertyDialogTabExpendyDialogbabe:FactePonAddAyDiabogTabEx  
(C++ function), 204 (C++ function), 206

sh::filepropertydialogtabs::FilePropertyDialogTabExpendyDialogbabe:FactePonAddAyDiabogTabEx  
(C++ function), 203 (C++ function), 205

sh::filepropertydialogtabs::FilePropertyDialogTabExpendyDialogbabe:FactePonAddAyDiabogTabEx  
(C++ function), 204 (C++ function), 205

sh::filepropertydialogtabs::FilePropertyDialogTabExpendyDialogbabe:FactePonAddAyDiabogTabEx  
(C++ function), 203 (C++ function), 204

sh::filepropertydialogtabs::FilePropertyDialogTabExpendyDialogbabe:FactePonAddAyDiabogTabEx  
(C++ function), 203 (C++ function), 205

sh::filepropertydialogtabs::FilePropertyDialogTabExpendyDialogbabe:FactePonAddAyDiabogTabEx  
(C++ function), 203 (C++ function), 205

sh::filepropertydialogtabs::FilePropertyDialogTabExpendyDialogbabe:FactePonAddAyDiabogTabEx  
(C++ function), 204 (C++ function), 205

sh::filepropertydialogtabs::FilePropertyDialogTabExpendyDialogbabe:FactePonAddAyDiabogTabEx  
(C++ function), 203 (C++ function), 205

sh::filepropertydialogtabs::FilePropertyDialogTabExpendyDialogbabe:FactePonAddAyDiabogTabEx  
(C++ function), 203 (C++ function), 205

sh::filepropertydialogtabs::FilePropertyDialogTabExpendyDialogbabe:FactePonAddAyDiabogTabEx  
(C++ function), 204 (C++ function), 205

sh::filepropertydialogtabs::FilePropertyDialogTabExpendyDialogbabe:FactePonAddAyDiabogTabEx  
(C++ function), 203 (C++ function), 205

sh::filepropertydialogtabs::FilePropertyDialogTabExpendyDialogbabe:FactePonAddAyDiabogTabEx  
(C++ function), 204 (C++ function), 205









(C++ member), 218, 775, 1381	function), 218, 774, 1381
sh::filesystem::Eurl::_cache_hostname (C++ member), 218, 775, 1381	sh::filesystem::Eurl::doShutdown (C++ function), 218, 775, 1381
sh::filesystem::Eurl::_cache_outermost_inshare (C++ member), 218, 775, 1381	sh::filesystem::Eurl::enwrapWithOuterUrl (C++ function), 217, 774, 1380
sh::filesystem::Eurl::_cache_outerurl (C++ member), 218, 775, 1381	sh::filesystem::Eurl::Eurl (C++ function), 218, 774, 1381
sh::filesystem::Eurl::_cache_outerurl_isthis (C++ member), 218, 775, 1381	sh::filesystem::Eurl::filenameCheck (C++ function), 218, 774, 1381
sh::filesystem::Eurl::_cache_parentsegment (C++ member), 218, 775, 1381	sh::filesystem::Eurl::FORBIDDEN_FILENAME_CHARACTER (C++ member), 218, 775, 1381
sh::filesystem::Eurl::_cache_path (C++ member), 218, 775, 1381	sh::filesystem::Eurl::fromString (C++ function), 218, 774, 1381
sh::filesystem::Eurl::_cache_root (C++ member), 218, 775, 1381	sh::filesystem::Eurl::fromStringNOCHECK (C++ function), 219, 775, 1382
sh::filesystem::Eurl::_cache_root_isthis (C++ member), 218, 775, 1381	sh::filesystem::Eurl::hasInnerUrls (C++ function), 217, 774, 1380
sh::filesystem::Eurl::_cache_scheme (C++ member), 219, 775, 1382	sh::filesystem::Eurl::hasParentSegment (C++ function), 217, 774, 1380
sh::filesystem::Eurl::_escape (C++ function), 219, 775, 1382	sh::filesystem::Eurl::hostname (C++ function), 217, 773, 1380
sh::filesystem::Eurl::_escapemap (C++ member), 219, 776, 1382	sh::filesystem::Eurl::isPrefixOf (C++ function), 218, 774, 1381
sh::filesystem::Eurl::_escapemapmutex (C++ member), 219, 776, 1382	sh::filesystem::Eurl::outermostInnerEurl (C++ function), 217, 773, 1380
sh::filesystem::Eurl::_eurlstring (C++ member), 218, 775, 1381	sh::filesystem::Eurl::outerUrl (C++ function), 217, 773, 1380
sh::filesystem::Eurl::_eurluniverse (C++ member), 219, 776, 1382	sh::filesystem::Eurl::outerUrlIsRootDirectory (C++ function), 217, 774, 1380
sh::filesystem::Eurl::_mutex (C++ member), 219, 776, 1382	sh::filesystem::Eurl::parentSegment (C++ function), 217, 774, 1380
sh::filesystem::Eurl::_unescape (C++ function), 219, 775, 1382	sh::filesystem::Eurl::path (C++ function), 217, 773, 1380
sh::filesystem::Eurl::~~Eurl (C++ function), 218, 774, 1381	sh::filesystem::Eurl::root (C++ function), 217, 774, 1380
sh::filesystem::Eurl::asString (C++ function), 217, 773, 1380	sh::filesystem::Eurl::scheme (C++ function), 217, 773, 1380
sh::filesystem::Eurl::basename (C++ function), 217, 773, 1380	sh::filesystem::Eurl::withAppendedSegment (C++ function), 217, 773, 1380
sh::filesystem::Eurl::check_filename (C++ function), 219, 775, 1382	sh::filesystem::Eurl::withAppendedSegments (C++ function), 217, 774, 1380
sh::filesystem::Eurl::check_hostname (C++ function), 219, 775, 1382	sh::filesystem::Eurl::WRAPPER_BEGIN (C++ member), 218, 775, 1381
sh::filesystem::Eurl::check_inner (C++ function), 219, 775, 1382	sh::filesystem::Eurl::WRAPPER_END (C++ member), 218, 775, 1381
sh::filesystem::Eurl::check_path (C++ function), 219, 775, 1382	sh::filesystem::EurlMisformattedException (C++ class), 219, 776, 1382
sh::filesystem::Eurl::check_scheme (C++ function), 219, 775, 1382	sh::filesystem::EurlMisformattedException::_demand (C++ function), 220, 777, 1383
sh::filesystem::Eurl::create (C++ function), 218, 774, 1381	sh::filesystem::EurlMisformattedException::autoRet (C++ function), 220, 776, 1383
sh::filesystem::Eurl::createNOCHECK (C++ function), 219, 775, 1382	sh::filesystem::EurlMisformattedException::auxiliary (C++ function), 219, 776, 1382
sh::filesystem::Eurl::doInitialize (C++	sh::filesystem::EurlMisformattedException::callsta



<b>Index</b>	<b>1945</b>
--------------	-------------

(C++ function), 223, 779, 1386  
 sh::filesystem::FilesystemHandler::itemIsExtendedAttribute (C++ function), 221, 777, 1384  
 sh::filesystem::FilesystemHandler::listExtendedAttributes (C++ function), 221, 778, 1384  
 sh::filesystem::FilesystemHandler::modelsh::filesystem::FilesystemModel::columnCount (C++ function), 223, 780, 1386  
 sh::filesystem::FilesystemHandler::removeExtendedAttribute (C++ function), 221, 778, 1384  
 sh::filesystem::FilesystemHandler::renameItem (C++ function), 222, 779, 1386  
 sh::filesystem::FilesystemHandler::requiresResolvedLinks (C++ function), 223, 779, 1386  
 sh::filesystem::FilesystemHandler::searchsh::filesystem::FilesystemModel::doInitialize (C++ function), 223, 779, 1386  
 sh::filesystem::FilesystemHandler::setCushonFilesystem (C++ function), 222, 778, 1385  
 sh::filesystem::FilesystemHandler::setExtendedAttribute (C++ function), 221, 778, 1384  
 sh::filesystem::FilesystemHandler::setMutex (C++ function), 221, 777, 1384  
 sh::filesystem::FilesystemHandler::viewExtendedSystem (C++ function), 223, 779, 1386  
 sh::filesystem::FilesystemHandler::viewLeftOfFilesystem (C++ function), 223, 779, 1386  
 sh::filesystem::FilesystemHandlerRegistersh::filesystem::FilesystemModel::findIndexesForEurl (C++ class), 224, 780, 1386  
 sh::filesystem::FilesystemHandlerRegistersh::addHandler (C++ function), 224, 780, 1387  
 sh::filesystem::FilesystemHandlerRegistersh::closeSystem (C++ function), 224, 780, 1387  
 sh::filesystem::FilesystemHandlerRegistersh::closeSystem (C++ function), 224, 780, 1387  
 sh::filesystem::FilesystemHandlerRegistersh::FilesystemHandlerRegisterModel::getOrCreateFilesystem (C++ function), 224, 781, 1387  
 sh::filesystem::FilesystemHandlerRegistersh::findHandler (C++ function), 224, 780, 1387  
 sh::filesystem::FilesystemHandlerRegistersh::handleSystem (C++ member), 224, 781, 1387  
 sh::filesystem::FilesystemHandlerRegistersh::isAlive (C++ function), 224, 780, 1387  
 sh::filesystem::FilesystemHandlerRegistersh::mutex (C++ member), 224, 781, 1387  
 sh::filesystem::FilesystemHandlerRegistersh::shutdown (C++ function), 224, 780, 1387  
 sh::filesystem::FilesystemModel (C++ class), 225, 781, 1387  
 sh::filesystem::FilesystemModel::\_findNodesForEasyshelper (C++ function), 227, 784, 1390  
 sh::filesystem::FilesystemModel::\_nodeDataMutex (C++ member), 227, 784, 1390  
 sh::filesystem::FilesystemModel::\_openNodeHelperMutex (C++ member), 228, 784, 1390  
 sh::filesystem::FilesystemModel::\_subitemFetchFlagsSystemChanged (C++ member), 227, 784, 1390  
 sh::filesystem::FilesystemModel::~FilesystemModel (C++ function), 227, 783, 1389  
 sh::filesystem::FilesystemModel::addOpenNodeHelper (C++ function), 227, 783, 1389  
 sh::filesystem::FilesystemModel::createFilesystemNode (C++ function), 226, 782, 1388  
 sh::filesystem::FilesystemModel::createFileviewProxy (C++ function), 225, 781, 1388  
 sh::filesystem::FilesystemModel::data (C++ function), 225, 781, 1387  
 sh::filesystem::FilesystemModel::doShutdown (C++ function), 227, 783, 1389  
 sh::filesystem::FilesystemModel::eurl2node (C++ member), 227, 784, 1390  
 sh::filesystem::FilesystemModel::eurl2nodemutex (C++ member), 227, 784, 1390  
 sh::filesystem::FilesystemModel::FilesystemModel (C++ function), 227, 784, 1390  
 sh::filesystem::FilesystemModel::findIndexesForEurl (C++ function), 225, 781, 1388  
 sh::filesystem::FilesystemModel::findNodesForEurl (C++ function), 225, 781, 1388  
 sh::filesystem::FilesystemModel::flags (C++ function), 225, 781, 1387  
 sh::filesystem::FilesystemModel::getIndexForNode (C++ function), 226, 782, 1388  
 sh::filesystem::FilesystemModel::getNodeForIndex (C++ function), 226, 782, 1388  
 sh::filesystem::FilesystemModel::getOrCreateFilesystem (C++ function), 226, 782, 1389  
 sh::filesystem::FilesystemModel::headerData (C++ function), 225, 781, 1387  
 sh::filesystem::FilesystemModel::index (C++ function), 225, 781, 1387  
 sh::filesystem::FilesystemModel::isAlive (C++ function), 227, 783, 1389  
 sh::filesystem::FilesystemModel::mutex (C++ member), 227, 784, 1390  
 sh::filesystem::FilesystemModel::openRootEurl (C++ function), 227, 784, 1390  
 sh::filesystem::FilesystemModel::parent (C++ function), 225, 781, 1387  
 sh::filesystem::FilesystemModel::refreshData (C++ function), 227, 783, 1389  
 sh::filesystem::FilesystemModel::rootNode (C++ function), 225, 781, 1387  
 sh::filesystem::FilesystemModel::rootnode (C++ member), 227, 784, 1390  
 sh::filesystem::FilesystemModel::rowCount (C++ member), 227, 784, 1390



(C++ function), 231, 787, 1393	(C++ function), 231, 787, 1393
sh::filesystem::FilesystemNode::childNodes	sh::filesystem::FilesystemNode::parentnode
(C++ function), 231, 787, 1393	(C++ function), 231, 788, 1394
sh::filesystem::FilesystemNode::childNodes	sh::filesystem::FilesystemNode::removeChild
(C++ function), 231, 787, 1393	(C++ function), 231, 788, 1394
sh::filesystem::FilesystemNode::DetailsRequest	sh::filesystem::FilesystemNode::removed
(C++ class), 235	(C++ function), 233, 789, 1395
sh::filesystem::FilesystemNode::DetailsRequest	sh::filesystem::FilesystemNode::requestContainedItem
(C++ member), 236	(C++ function), 232, 789, 1395
sh::filesystem::FilesystemNode::DetailsRequest	sh::filesystem::FilesystemNode::requestDetails
(C++ function), 236	(C++ function), 232, 788, 1394
sh::filesystem::FilesystemNode::DetailsRequest	sh::filesystem::FilesystemNode::setDetail
(C++ member), 236	(C++ function), 233, 789, 1395
sh::filesystem::FilesystemNode::DetailsRequest	sh::filesystem::FilesystemNode::setDisplayName
(C++ member), 236	(C++ function), 231, 787, 1393
sh::filesystem::FilesystemNode::DetailsRequest	sh::filesystem::FilesystemNode::setHidden
(C++ function), 236	(C++ function), 232, 788, 1394
sh::filesystem::FilesystemNode::displayName	sh::filesystem::FilesystemNode::setIcon
(C++ function), 231, 787, 1393	(C++ function), 232, 788, 1394
sh::filesystem::FilesystemNode::eurl	sh::filesystem::FilesystemNode::setLinkDestination
(C++ function), 231, 787, 1393	(C++ function), 232, 788, 1394
sh::filesystem::FilesystemNode::FilesystemNode	sh::filesystem::FilesystemNodeList (C++
(C++ function), 232, 789, 1395	class), 233, 789, 1395
sh::filesystem::FilesystemNode::getDetails	sh::filesystem::FilesystemNodeList::addItem
(C++ function), 232, 788, 1394	(C++ function), 233, 790, 1396
sh::filesystem::FilesystemNode::getDetails	sh::filesystem::FilesystemNodeList::addItem
(C++ function), 232, 788, 1394	(C++ function), 233, 790, 1396
sh::filesystem::FilesystemNode::getInsertPos	sh::filesystem::FilesystemNodeList::contains
(C++ function), 233, 789, 1395	(C++ function), 234, 790, 1396
sh::filesystem::FilesystemNode::handler	sh::filesystem::FilesystemNodeList::mynode
(C++ function), 231, 787, 1393	(C++ function), 234, 790, 1396
sh::filesystem::FilesystemNode::icon	sh::filesystem::FilesystemNodeList::nodes
(C++ function), 232, 788, 1394	(C++ function), 234, 790, 1396
sh::filesystem::FilesystemNode::iconChanged	sh::filesystem::FilesystemNodeList::removeItem
(C++ function), 233, 789, 1395	(C++ function), 233, 790, 1396
sh::filesystem::FilesystemNode::index	sh::filesystem::FilesystemNodeList::removeItems
(C++ function), 231, 788, 1394	(C++ function), 233, 790, 1396
sh::filesystem::FilesystemNode::isAlive	sh::filesystem::FilesystemNodeList::resetItems
(C++ function), 232, 788, 1394	(C++ function), 234, 790, 1396
sh::filesystem::FilesystemNode::isConfigured	sh::filesystem::FilesystemNodeListEditor
(C++ function), 232, 788, 1394	(C++ class), 234, 790, 1396
sh::filesystem::FilesystemNode::isFetching	sh::filesystem::FilesystemNodeListEditor::_beganAdd
(C++ function), 231, 788, 1394	(C++ member), 235, 791, 1397
sh::filesystem::FilesystemNode::isHidden	sh::filesystem::FilesystemNodeListEditor::_handler
(C++ function), 232, 788, 1394	(C++ member), 235, 791, 1397
sh::filesystem::FilesystemNode::isHiddenChanged	sh::filesystem::FilesystemNodeListEditor::_itemsAre
(C++ function), 233, 789, 1395	(C++ member), 235, 791, 1397
sh::filesystem::FilesystemNode::isRootNode	sh::filesystem::FilesystemNodeListEditor::_iterativ
(C++ function), 232, 788, 1394	(C++ member), 235, 792, 1398
sh::filesystem::FilesystemNode::linkDestination	sh::filesystem::FilesystemNodeListEditor::_iterativ
(C++ function), 232, 788, 1394	(C++ member), 235, 791, 1397
sh::filesystem::FilesystemNode::model	sh::filesystem::FilesystemNodeListEditor::_list
(C++ function), 231, 787, 1393	(C++ member), 235, 791, 1397
sh::filesystem::FilesystemNode::nodetypes	sh::filesystem::FilesystemNodeListEditor::_model



(C++ member), 235, 791, 1397  
 sh::filesystem::FilesystemNodeListEditorsh::node\_typeem::FilesystemOperation  
 (C++ member), 235, 791, 1397 (C++ class), 236, 792, 1398  
 sh::filesystem::FilesystemNodeListEditorsh::parentsh::FilesystemOperation::\_detailColumn  
 (C++ member), 235, 791, 1397 (C++ member), 240, 795, 1401  
 sh::filesystem::FilesystemNodeListEditorsh::parentsh::FilesystemOperation::\_detailColumn  
 (C++ member), 235, 791, 1397 (C++ member), 240, 795, 1401  
 sh::filesystem::FilesystemNodeListEditorsh::pendingSystematFileAddingOperation::\_detailColumn  
 (C++ member), 235, 792, 1398 (C++ member), 240, 795, 1401  
 sh::filesystem::FilesystemNodeListEditorsh::pendingMutexFilesystemOperation::\_handler  
 (C++ member), 235, 792, 1398 (C++ function), 239, 795, 1401  
 sh::filesystem::FilesystemNodeListEditorsh::FilesystemNodeListEditorOperation::\_monitor  
 (C++ function), 234, 790, 1396 (C++ function), 239, 795, 1401  
 sh::filesystem::FilesystemNodeListEditorshaddFilesystemem::FilesystemOperation::\_operation  
 (C++ function), 234, 790, 1396 (C++ member), 240, 795, 1401  
 sh::filesystem::FilesystemNodeListEditorshaddExistingNodeFilesystemOperation::\_resolveIfNeed  
 (C++ function), 234, 791, 1397 (C++ function), 239, 795, 1401  
 sh::filesystem::FilesystemNodeListEditorshaddFilesystemem::FilesystemOperation::addTransferrak  
 (C++ function), 234, 790, 1396 (C++ function), 239, 795, 1401  
 sh::filesystem::FilesystemNodeListEditorshbeginFilesystemem::FilesystemOperation::canCopyItem  
 (C++ function), 234, 790, 1396 (C++ function), 238, 794, 1400  
 sh::filesystem::FilesystemNodeListEditorshendFilesystemem::FilesystemOperation::canCreateDirec  
 (C++ function), 234, 791, 1397 (C++ function), 236, 792, 1398  
 sh::filesystem::FilesystemNodeListEditorshFilesystemNodeListEditorOperation::canCreateFile  
 (C++ function), 234, 790, 1396 (C++ function), 237, 793, 1399  
 sh::filesystem::FilesystemNodeListEditorshgetChesystemem::FilesystemOperation::canCreateLink  
 (C++ function), 235, 791, 1397 (C++ function), 237, 793, 1398  
 sh::filesystem::FilesystemNodeListEditorshrawFilesystemem::FilesystemOperation::canDeleteItem  
 (C++ function), 235, 791, 1397 (C++ function), 237, 793, 1399  
 sh::filesystem::FilesystemNodeListEditorshseFilesystemem::FilesystemOperation::canGetFileCont  
 (C++ function), 234, 790, 1396 (C++ function), 236, 792, 1398  
 sh::filesystem::FilesystemNodeListEditorshseFilesystemem::FilesystemOperation::canMoveItem  
 (C++ function), 235, 791, 1397 (C++ function), 237, 793, 1399  
 sh::filesystem::FilesystemNodeType (C++ sh::filesystem::FilesystemOperation::copyItem  
 enum), 769, 1375 (C++ function), 238, 794, 1400  
 sh::filesystem::FilesystemNodeType::DireshorFilesystemem::FilesystemOperation::copyItems  
 (C++ enumerator), 769, 1376 (C++ function), 238, 794, 1400  
 sh::filesystem::FilesystemNodeType::Filesh::filesystem::FilesystemOperation::createDirector  
 (C++ enumerator), 769, 1376 (C++ function), 237, 792, 1398  
 sh::filesystem::FilesystemNodeType::FIRSTTYPEFilesystemem::FilesystemOperation::createFile  
 (C++ enumerator), 769, 1376 (C++ function), 237, 793, 1399  
 sh::filesystem::FilesystemNodeType::Invalidsh::filesystem::FilesystemOperation::createLink  
 (C++ enumerator), 769, 1376 (C++ function), 237, 793, 1398  
 sh::filesystem::FilesystemNodeType::LASTBHYSCALSYBEm::FilesystemOperation::deleteDirector  
 (C++ enumerator), 769, 1376 (C++ function), 237, 793, 1399  
 sh::filesystem::FilesystemNodeType::LASTTYPEFilesystemem::FilesystemOperation::deleteItem  
 (C++ enumerator), 769, 1376 (C++ function), 237, 793, 1399  
 sh::filesystem::FilesystemNodeType::Linksh::filesystem::FilesystemOperation::FilesystemOper  
 (C++ enumerator), 769, 1376 (C++ function), 236, 792, 1398  
 sh::filesystem::FilesystemNodeType::NONEsh::filesystem::FilesystemOperation::getExtendedAtt  
 (C++ enumerator), 769, 1375 (C++ function), 239, 794, 1400  
 sh::filesystem::FilesystemNodeType::Specsh::Filesystemem::FilesystemOperation::getExtendedAtt  
 (C++ enumerator), 769, 1376 (C++ function), 239, 794, 1400  
 sh::filesystem::FilesystemNodeType::Unknownsh::filesystem::FilesystemOperation::getFileContent

[illegible]



[illegible]





[illegible]



[illegible]



Index	1957
-------	------

```

(C++ function), 285, 821, 822, 1427, 1428
sh::filesystem::Operation (C++ class), 286, 822, 1428
sh::filesystem::Operation::_customData (C++ member), 287, 824, 1430
sh::filesystem::Operation::_detailsCache (C++ member), 287, 824, 1430
sh::filesystem::Operation::_detailsCacheEnabled (C++ member), 287, 824, 1430
sh::filesystem::Operation::_filesystemOperations (C++ member), 287, 824, 1430
sh::filesystem::Operation::_maxAllowedSizeRatioPerFilesystem (C++ member), 287, 824, 1430
sh::filesystem::Operation::_tempdir (C++ member), 288, 824, 1430
sh::filesystem::Operation::_tempfilemutex (C++ member), 288, 824, 1430
sh::filesystem::Operation::~Operation (C++ function), 287, 823, 1429
sh::filesystem::Operation::~abort (C++ function), 286, 823, 1428
sh::filesystem::Operation::~commit (C++ function), 286, 823, 1429
sh::filesystem::Operation::~enableDetailsCache (C++ function), 286, 823, 1429
sh::filesystem::Operation::~fetchContainersFile (C++ function), 286, 822, 1428
sh::filesystem::Operation::~fetchFile (C++ function), 286, 822, 823, 1428
sh::filesystem::Operation::~fileIsTemporary (C++ member), 287, 824, 1430
sh::filesystem::Operation::~files (C++ member), 287, 824, 1430
sh::filesystem::Operation::~filesystem (C++ function), 287, 823, 1429
sh::filesystem::Operation::~getCustomData (C++ function), 287, 823, 1429
sh::filesystem::Operation::~getDetailFromCache (C++ function), 287, 823, 1429
sh::filesystem::Operation::~getFreeCacheSpace (C++ function), 286, 822, 1428
sh::filesystem::Operation::~getTempDir (C++ function), 287, 823, 1429
sh::filesystem::Operation::isEnabledDetailsCache (C++ function), 287, 823, 1429
sh::filesystem::Operation::MaxAllowedSizeRatioPerFilesystem (C++ class), 288, 289, 824, 1430
sh::filesystem::Operation::MaxAllowedSizeRatioPerFilesystemExceededException (C++ member), 289, 290, 825, 1431
sh::filesystem::Operation::MaxAllowedSizeRatioPerFilesystemExceededException::operator demangle (C++ function), 286, 822, 1428
sh::filesystem::Operation::MaxAllowedSizeRatioPerFilesystemExceededException::operator bool (C++ member), 287, 824, 1430
sh::filesystem::Operation::MaxAllowedSizeRatioPerFilesystemExceededException::pendingCommit (C++ function), 287, 823, 1429
sh::filesystem::Operation::MaxAllowedSizeRatioPerFilesystemExceededException::setCustomData (C++ function), 287, 823, 1429

```



sh::filesystemhandlers::ActionMountNetworkGnomeIOSystemhandlersCheckedUnmountGnomeIOLocation  
(C++ function), 293, 828, 1434 (C++ function), 295, 829, 1436

sh::filesystemhandlers::ActionMountNetworkGnomeIOSystemhandlersEnabledUnmountGnomeIOLocation  
(C++ function), 293, 828, 1434 (C++ function), 294, 829, 1435

sh::filesystemhandlers::ActionMountNetworkGnomeIOSystemhandlersIsActionUnmountGnomeIOLocation  
(C++ function), 293, 828, 1434 (C++ function), 294, 829, 1435

sh::filesystemhandlers::ActionMountNetworkGnomeIOSystemhandlersIsShActionHimountGnomeIOLocation  
(C++ function), 292, 827, 1433 (C++ function), 294, 829, 1435

sh::filesystemhandlers::ActionMountNetworkGnomeIOSystemhandlersIsTeActionUnmountGnomeIOLocation  
(C++ function), 293, 828, 1434 (C++ function), 294, 829, 1435

sh::filesystemhandlers::ActionMountNetworkGnomeIOSystemhandlersIsViaCblenUnmountGnomeIOLocation  
(C++ function), 293, 828, 1434 (C++ function), 294, 829, 1435

sh::filesystemhandlers::ActionMountNetworkGnomeIOSystemhandlersIsShortAuthonUnmountGnomeIOLocation  
(C++ function), 292, 827, 1433 (C++ function), 294, 829, 1436

sh::filesystemhandlers::ActionMountNetworkGnomeIOSystemhandlersIsShortAuthonUffmogggesomewolenaDio  
(C++ function), 292, 827, 1433 (C++ function), 294, 829, 1435

sh::filesystemhandlers::ActionMountNetworkGnomeIOSystemhandlersIsText:ActionUnmountGnomeIOLocation  
(C++ function), 292, 827, 1434 (C++ function), 294, 829, 1435

sh::filesystemhandlers::ActionMountNetworkGnomeIOSystemhandlersIsVisibleActionUnmountGnomeIOLocation  
(C++ function), 293, 828, 1434 (C++ function), 294, 829, 1435

sh::filesystemhandlers::ActionUnmountGnomeIOLocationSystemhandlers::ActionUnmountGnomeIOLocation  
(C++ class), 294, 828, 1435 (C++ function), 295, 830, 1436

sh::filesystemhandlers::ActionUnmountGnomeIOLocationSystemhandlers::ParentActionUnmountGnomeIOLocation  
(C++ function), 295, 830, 1436 (C++ function), 295, 829, 1436

sh::filesystemhandlers::ActionUnmountGnomeIOLocationSystemhandlers::AndbeVsmoAntGnomeIOSystemHandler  
(C++ function), 294, 829, 1435 (C++ class), 295, 830, 1436

sh::filesystemhandlers::ActionUnmountGnomeIOLocationSystemhandlers::ArchiveFilesystemHandler::A  
(C++ function), 295, 830, 1436 (C++ class), 299

sh::filesystemhandlers::ActionUnmountGnomeIOLocationSystemhandlers::ActAnoRiveEdesystemHandler::A  
(C++ function), 294, 829, 1435 (C++ function), 300

sh::filesystemhandlers::ActionUnmountGnomeIOLocationSystemhandlers::AerializeArchiveFilesystemHandler::A  
(C++ function), 295, 830, 1436 (C++ function), 299

sh::filesystemhandlers::ActionUnmountGnomeIOLocationSystemhandlers::AshuttedownArchiveFilesystemHandler::A  
(C++ function), 295, 830, 1436 (C++ function), 301

sh::filesystemhandlers::ActionUnmountGnomeIOLocationSystemhandlers::ArchiveFilesystemHandler::A  
(C++ function), 294, 829, 1435 (C++ function), 300

sh::filesystemhandlers::ActionUnmountGnomeIOLocationSystemhandlers::ArchiveFilesystemHandler::A  
(C++ function), 294, 829, 1435 (C++ function), 300

sh::filesystemhandlers::ActionUnmountGnomeIOLocationSystemhandlers::ArchiveFilesystemHandler::A  
(C++ function), 294, 829, 1435 (C++ function), 300

sh::filesystemhandlers::ActionUnmountGnomeIOLocationSystemhandlers::ArchiveFilesystemHandler::A  
(C++ function), 294, 829, 1435 (C++ function), 299

sh::filesystemhandlers::ActionUnmountGnomeIOLocationSystemhandlers::ArchiveFilesystemHandler::A  
(C++ function), 295, 830, 1436 (C++ function), 299

sh::filesystemhandlers::ActionUnmountGnomeIOLocationSystemhandlers::ArchiveFilesystemHandler::A  
(C++ function), 295, 830, 1436 (C++ function), 300

sh::filesystemhandlers::ActionUnmountGnomeIOLocationSystemhandlers::ArchiveFilesystemHandler::A  
(C++ function), 294, 829, 1435 (C++ function), 300

sh::filesystemhandlers::ActionUnmountGnomeIOLocationSystemhandlers::ArchiveFilesystemHandler::A  
(C++ function), 294, 829, 1435 (C++ function), 300

sh::filesystemhandlers::ActionUnmountGnomeIOLocationSystemhandlers::ArchiveFilesystemHandler::A  
(C++ function), 295, 830, 1436 (C++ function), 300

sh::filesystemhandlers::ActionUnmountGnomeIOLocationSystemhandlers::ArchiveFilesystemHandler::A  
(C++ function), 295, 830, 1436 (C++ function), 300

sh::filesystemhandlers::ActionUnmountGnomeIOLocationSystemhandlers::ArchiveFilesystemHandler::A  
(C++ function), 295, 830, 1436 (C++ function), 300

sh::filesystemhandlers::ActionUnmountGnomeIOLocationSystemhandlers::ArchiveFilesystemHandler::A  
(C++ function), 295, 830, 1436 (C++ function), 300





```

sh::filesystemhandlers::ArchiveFilesystemHandler::ArchiveFilesystemHandler::
(C++ function), 296, 298, 831, 833, 1437, (C++ function), 298, 833, 1439
1439 sh::filesystemhandlers::ArchiveFilesystemHandler::ArchiveFilesystemHandler::
sh::filesystemhandlers::ArchiveFilesystemHandler(C++ function), 299, 834, 1440
(C++ function), 299, 834, 1440 sh::filesystemhandlers::ArchiveFilesystemHandler::
sh::filesystemhandlers::ArchiveFilesystemHandler(C++ function), 297, 832, 1438
(C++ function), 298, 833, 1439 sh::filesystemhandlers::ArchiveFilesystemHandler::
sh::filesystemhandlers::ArchiveFilesystemHandler(C++ function), 297, 832, 1438
(C++ function), 296, 298, 831, 833, 1437, sh::filesystemhandlers::ArchiveFilesystemHandler::
1439 (C++ function), 296, 297, 830, 832, 1437,
sh::filesystemhandlers::ArchiveFilesystemHandler::doInitialize
(C++ function), 299, 834, 1440 sh::filesystemhandlers::ArchiveFilesystemHandler::
sh::filesystemhandlers::ArchiveFilesystemHandler(C++ function), 298, 833, 1439
(C++ function), 299, 834, 1440 sh::filesystemhandlers::ArchiveFilesystemHandler::
sh::filesystemhandlers::ArchiveFilesystemHandler(C++ function), 298, 833, 1439
(C++ function), 296, 298, 831, 833, 1437, sh::filesystemhandlers::GnomeIODevicesFilesystemHan
1439 (C++ class), 302, 834, 1440
sh::filesystemhandlers::ArchiveFilesystemHandler::systemHandlerAttributeGnomeIODevicesFilesystemHan
(C++ function), 297, 832, 1438 (C++ function), 302, 834, 1440
sh::filesystemhandlers::ArchiveFilesystemHandler::systemHandlerAttributeGnomeIODevicesFilesystemHan
(C++ function), 297, 832, 1438 (C++ function), 303, 835, 1441
sh::filesystemhandlers::ArchiveFilesystemHandler::systemHandlerAttributeGnomeIODevicesFilesystemHan
(C++ function), 297, 832, 1438 (C++ function), 304, 836, 1442
sh::filesystemhandlers::ArchiveFilesystemHandler::systemHandlerAttributeGnomeIODevicesFilesystemHan
(C++ function), 296, 297, 831, 832, 1437, (C++ function), 304, 836, 1442
1438 sh::filesystemhandlers::GnomeIODevicesFilesystemHan
sh::filesystemhandlers::ArchiveFilesystemHandler(C++ function), 304, 836, 1442
(C++ function), 296, 297, 831, 832, 1437, sh::filesystemhandlers::GnomeIODevicesFilesystemHan
1438 (C++ function), 303, 835, 1441
sh::filesystemhandlers::ArchiveFilesystemHandler::systemHandlerTypes::GnomeIODevicesFilesystemHan
(C++ function), 296, 297, 830, 832, 1437, (C++ function), 304, 836, 1442
1438 sh::filesystemhandlers::GnomeIODevicesFilesystemHan
sh::filesystemhandlers::ArchiveFilesystemHandler(C++ function), 305, 837, 1443
(C++ function), 296, 297, 830, 831, 1436, sh::filesystemhandlers::GnomeIODevicesFilesystemHan
1438 (C++ function), 303, 304, 835, 1441
sh::filesystemhandlers::ArchiveFilesystemHandler::systemHandlers::GnomeIODevicesFilesystemHan
(C++ function), 296, 297, 830, 831, 1436, (C++ function), 304, 836, 1442
1437 sh::filesystemhandlers::GnomeIODevicesFilesystemHan
sh::filesystemhandlers::ArchiveFilesystemHandler(C++ function), 304, 836, 1442
(C++ function), 296, 297, 830, 831, 1436, sh::filesystemhandlers::GnomeIODevicesFilesystemHan
1437 (C++ function), 306, 838, 1444
sh::filesystemhandlers::ArchiveFilesystemHandler::systemHandlerKnownSchemeGnomeIODevicesFilesystemHan
(C++ function), 299, 833, 1439 (C++ function), 305, 837, 1443
sh::filesystemhandlers::ArchiveFilesystemHandler::systemhandlers::GnomeIODevicesFilesystemHan
(C++ function), 296, 830, 831, 1436, 1437 (C++ function), 304, 836, 1442
sh::filesystemhandlers::ArchiveFilesystemHandler::systemHandlerAttributeGnomeIODevicesFilesystemHan
(C++ function), 297, 832, 1438 (C++ function), 306, 838, 1444
sh::filesystemhandlers::ArchiveFilesystemHandler::systemhandlers::GnomeIODevicesFilesystemHan
(C++ function), 299, 834, 1440 (C++ function), 306, 838, 1444
sh::filesystemhandlers::ArchiveFilesystemHandler::systemHandlerExtendedAttributeGnomeIODevicesFilesystemHan
(C++ function), 297, 832, 1438 (C++ function), 306, 838, 1444
sh::filesystemhandlers::ArchiveFilesystemHandler::systemHandlerItems::GnomeIODevicesFilesystemHan
(C++ function), 296, 298, 831, 833, 1437, (C++ function), 302, 304, 836, 1440,
1439 1442

```

sh::filesystemhandlers::GnomeIODevicesFilesystemHandler::castGnomeIODevicesFilesystemHandler  
 (C++ function), 305, 837, 1443 (C++ function), 304, 836, 1442  
 sh::filesystemhandlers::GnomeIODevicesFilesystemHandler::extendedAttributesFilesystemHandler  
 (C++ function), 305, 837, 1443 (C++ function), 302, 834, 1440  
 sh::filesystemhandlers::GnomeIODevicesFilesystemHandler::extendedAttributesFilesystemHandler  
 (C++ function), 305, 837, 1443 (C++ function), 305, 837, 1443  
 sh::filesystemhandlers::GnomeIODevicesFilesystemHandler::fileGnomeIODevicesFilesystemHandler  
 (C++ function), 303, 835, 1441 (C++ function), 305, 838, 1443  
 sh::filesystemhandlers::GnomeIODevicesFilesystemHandler::linkGnomeIODevicesFilesystemHandler  
 (C++ function), 303, 835, 1441 (C++ function), 305, 837, 1443  
 sh::filesystemhandlers::GnomeIODevicesFilesystemHandler::mimeGnomeIODevicesFilesystemHandler  
 (C++ function), 303, 835, 1441 (C++ function), 305, 837, 1443  
 sh::filesystemhandlers::GnomeIODevicesFilesystemHandler::mimeGnomeIODevicesFilesystemHandler  
 (C++ function), 303, 835, 1441 (C++ function), 303, 835, 1441  
 sh::filesystemhandlers::GnomeIODevicesFilesystemHandler::sizeGnomeIODevicesFilesystemHandler  
 (C++ function), 303, 835, 1441 (C++ function), 306, 838, 1444  
 sh::filesystemhandlers::GnomeIODevicesFilesystemHandler::typeGnomeIODevicesFilesystemHandler  
 (C++ function), 302, 303, 834, 1440 (C++ function), 306, 838, 1444  
 sh::filesystemhandlers::GnomeIODevicesFilesystemHandler::GnomeIODevicesFilesystemHandler  
 (C++ struct), 1880 (C++ function), 306, 838, 1444  
 sh::filesystemhandlers::GnomeIODevicesFilesystemHandler::GnomeIODevicesFilesystemHandler  
 (C++ member), 1881 (C++ function), 305, 837, 1443  
 sh::filesystemhandlers::GnomeIODevicesFilesystemHandler::GnomeIODevicesFilesystemHandler  
 (C++ function), 1881 (C++ function), 305, 837, 1443  
 sh::filesystemhandlers::GnomeIODevicesFilesystemHandler::GnomeIODevicesFilesystemHandler  
 (C++ member), 1881 (C++ class), 307, 839, 1444  
 sh::filesystemhandlers::GnomeIODevicesFilesystemHandler::GnomeIODevicesFilesystemHandler:::  
 (C++ member), 1881 (C++ function), 307, 309, 839, 841, 1445,  
 sh::filesystemhandlers::GnomeIODevicesFilesystemHandler::GnomeIODevice::url  
 (C++ member), 1881 sh::filesystemhandlers::GnomeIOFilesystemHandler:::  
 sh::filesystemhandlers::GnomeIODevicesFilesystemHandler::GnomeIODevice::url  
 (C++ function), 306, 838, 1444 1447  
 sh::filesystemhandlers::GnomeIODevicesFilesystemHandler::GnomeIOFilesystemHandler:::  
 (C++ function), 306, 838, 1444 (C++ function), 307, 309, 839, 841, 1445,  
 sh::filesystemhandlers::GnomeIODevicesFilesystemHandler::isAlive  
 (C++ function), 306, 838, 1444 sh::filesystemhandlers::GnomeIOFilesystemHandler:::  
 sh::filesystemhandlers::GnomeIODevicesFilesystemHandler::isAlive  
 (C++ function), 306, 838, 1444 sh::filesystemhandlers::GnomeIOFilesystemHandler:::  
 sh::filesystemhandlers::GnomeIODevicesFilesystemHandler::isAlive  
 (C++ function), 305, 837, 1443 1447  
 sh::filesystemhandlers::GnomeIODevicesFilesystemHandler::isAlive  
 (C++ function), 302, 303, 834, 1440 (C++ function), 307, 309, 839, 841, 1445,  
 sh::filesystemhandlers::GnomeIODevicesFilesystemHandler::listExtendedAttributes  
 (C++ function), 305, 837, 1443 sh::filesystemhandlers::GnomeIOFilesystemHandler:::  
 sh::filesystemhandlers::GnomeIODevicesFilesystemHandler::listExtendedAttributes  
 (C++ function), 306, 838, 1444 1447  
 sh::filesystemhandlers::GnomeIODevicesFilesystemHandler::GnomeIOFilesystemHandler:::  
 (C++ function), 306, 838, 1444 (C++ function), 308, 840, 1446  
 sh::filesystemhandlers::GnomeIODevicesFilesystemHandler::GnomeIOFilesystemHandler:::  
 (C++ function), 306, 838, 1444 (C++ function), 307, 309, 839, 841, 1445,  
 sh::filesystemhandlers::GnomeIODevicesFilesystemHandler::mountLocation  
 (C++ function), 306, 838, 1444 sh::filesystemhandlers::GnomeIOFilesystemHandler:::  
 sh::filesystemhandlers::GnomeIODevicesFilesystemHandler::mountLocation  
 (C++ function), 302, 834, 1440 1447  
 sh::filesystemhandlers::GnomeIODevicesFilesystemHandler::moveGnomeIOFilesystemHandler:::  
 (C++ function), 305, 837, 1443 (C++ function), 307, 309, 839, 841, 1445,

```

1447 sh::filesystemhandlers::GnomeIOFilesystemHandler::
sh::filesystemhandlers::GnomeIOFilesystemHandler(C++ function), 308, 840, 1446
(C++ function), 310, 842, 1448 sh::filesystemhandlers::GnomeIOFilesystemHandler::
sh::filesystemhandlers::GnomeIOFilesystemHandler(C++ function), 310, 843, 1448 empty
(C++ function), 309, 842, 1447 sh::filesystemhandlers::GnomeIOFilesystemHandler::n
sh::filesystemhandlers::GnomeIOFilesystemHandler(C++ function), 310, 842, 1448
(C++ function), 307, 309, 840, 842, 1445, sh::filesystemhandlers::GnomeIOFilesystemHandler::n
1447 (C++ function), 310, 843, 1448
sh::filesystemhandlers::GnomeIOFilesystemHandlersh::filesystemhandlers::GnomeIOFilesystemHandler::n
(C++ function), 310, 843, 1448 (C++ function), 310, 843, 1448
sh::filesystemhandlers::GnomeIOFilesystemHandlersh::filesystemhandlers::GnomeIOFilesystemHandler::M
(C++ function), 310, 843, 1448 (C++ class), 310, 311, 843, 1448
sh::filesystemhandlers::GnomeIOFilesystemHandlersh::filesystemhandlers::GnomeIOFilesystemHandler::M
(C++ function), 308, 310, 840, 842, 1445, (C++ member), 311, 312, 843, 1449
1448 sh::filesystemhandlers::GnomeIOFilesystemHandler::M
sh::filesystemhandlers::GnomeIOFilesystemHandler(C++ member), 311, 843, 1449
(C++ function), 308, 841, 1446 sh::filesystemhandlers::GnomeIOFilesystemHandler::M
sh::filesystemhandlers::GnomeIOFilesystemHandler(C++ member), 311, 843, 1449
(C++ function), 308, 841, 1446 sh::filesystemhandlers::GnomeIOFilesystemHandler::M
sh::filesystemhandlers::GnomeIOFilesystemHandler(C++ function), 311, 843, 1449
(C++ function), 308, 840, 1446 sh::filesystemhandlers::GnomeIOFilesystemHandler::M
sh::filesystemhandlers::GnomeIOFilesystemHandler(C++ function), 311, 843, 1449
(C++ function), 307, 309, 839, 841, 1445, sh::filesystemhandlers::GnomeIOFilesystemHandler::M
1447 (C++ struct), 311, 843, 1449, 1881
sh::filesystemhandlers::GnomeIOFilesystemHandlersh::filesystemhandlers::GnomeIOFilesystemHandler::M
(C++ function), 307, 308, 839, 840, 1445, (C++ member), 311, 843, 1449, 1881
1446 sh::filesystemhandlers::GnomeIOFilesystemHandler::M
sh::filesystemhandlers::GnomeIOFilesystemHandler(C++ member), 311, 843, 1449, 1881
(C++ function), 307, 308, 839, 840, 1445, sh::filesystemhandlers::GnomeIOFilesystemHandler::M
1446 (C++ member), 311, 843, 1449, 1881
sh::filesystemhandlers::GnomeIOFilesystemHandlersh::filesystemhandlers::GnomeIOFilesystemHandler::M
(C++ function), 307, 308, 839, 840, 1445, (C++ member), 311, 843, 1449, 1881
1446 sh::filesystemhandlers::GnomeIOFilesystemHandler::M
sh::filesystemhandlers::GnomeIOFilesystemHandler(C++ function), 311, 843, 1449, 1881
(C++ function), 307, 308, 839, 840, 1445, sh::filesystemhandlers::GnomeIOFilesystemHandler::M
1446 (C++ function), 308, 841, 1446
sh::filesystemhandlers::GnomeIOFilesystemHandlersh::filesystemhandlers::GnomeIOFilesystemHandler::M
(C++ function), 307, 308, 839, 840, 1445, (C++ function), 308, 309, 840, 842, 1445,
1446 1448
sh::filesystemhandlers::GnomeIOFilesystemHandlersh::filesystemhandlers::GnomeIOFilesystemHandler::M
(C++ function), 307, 839, 1445 (C++ function), 308, 840, 1445
sh::filesystemhandlers::GnomeIOFilesystemHandlersh::filesystemhandlers::GnomeIOFilesystemHandler::s
(C++ class), 311 (C++ function), 310, 842, 1448
sh::filesystemhandlers::GnomeIOFilesystemHandlersh::filesystemhandlers::GnomeIOFilesystemHandler::s
(C++ function), 311 (C++ function), 309, 841, 1447
sh::filesystemhandlers::GnomeIOFilesystemHandlersh::filesystemhandlers::GnomeIOFilesystemHandler::s
(C++ function), 311 (C++ function), 308, 841, 1446
sh::filesystemhandlers::GnomeIOFilesystemHandlersh::filesystemhandlers::GnomeIOFilesystemHandler::s
(C++ function), 310, 843, 1448 (C++ function), 307, 308, 839, 840, 1445,
sh::filesystemhandlers::GnomeIOFilesystemHandler1446isWellKnownScheme
(C++ function), 308, 840, 1446 sh::filesystemhandlers::GnomeIOFilesystemHandler::t
sh::filesystemhandlers::GnomeIOFilesystemHandler(C++ function), 310, 843, 1448
(C++ function), 307, 308, 839, 840, 1445, sh::filesystemhandlers::GnomeIOFilesystemHandler::u
1446 (C++ function), 310, 843, 1448

```









```

(C++ function), 325, 327, 857, 859, 1462,
1465
sh::filesystemhandlers::LocalFilesystemHandler::mo
sh::filesystemhandlers::LocalFilesystemHandler:(C++ function), 328, 860, 1466
(C++ function), 328, 860, 1466
sh::filesystemhandlers::LocalFilesystemHandler::re
sh::filesystemhandlers::LocalFilesystemHandler:(C++ function), 325, 326, 857, 859, 1463,
(C++ function), 327, 860, 1465
1464
sh::filesystemhandlers::LocalFilesystemHandler::re
sh::filesystemhandlers::LocalFilesystemHandler:(C++ function), 325, 327, 857, 859, 1462,
(C++ function), 325, 327, 857, 860, 1463,
1465
1465
sh::filesystemhandlers::LocalFilesystemHandler::re
sh::filesystemhandlers::LocalFilesystemHandler:(C++ function), 328, 860, 1466
(C++ function), 325, 858, 1463
sh::filesystemhandlers::LocalFilesystemHandler::se
sh::filesystemhandlers::LocalFilesystemHandler:(C++ function), 325, 858, 1463
(C++ function), 327, 860, 1465
sh::filesystemhandlers::LocalFilesystemHandler::se
sh::filesystemhandlers::LocalFilesystemHandler:(C++ function), 325, 327, 857, 860, 1463,
(C++ function), 325, 326, 858, 859, 1463,
1465
1464
sh::filesystemhandlers::LocalFilesystemHandler::se
sh::filesystemhandlers::LocalFilesystemHandler:(C++ function), 325, 326, 857, 859, 1463,
(C++ function), 325, 326, 857, 858, 1463,
1464
1464
sh::filesystemhandlers::LocalFilesystemHandler::se
sh::filesystemhandlers::LocalFilesystemHandler:(C++ function), 325, 326, 857, 858, 1463,
(C++ function), 328, 860, 1466
1464
sh::filesystemhandlers::LocalFilesystemHandler::vi
sh::filesystemhandlers::LocalFilesystemHandler:(C++ function), 327, 860, 1465
(C++ function), 324, 326, 857, 859, 1462,
sh::filesystemhandlers::LocalFilesystemHandler::vi
1464
(C++ function), 327, 860, 1465
sh::filesystemhandlers::LocalFilesystemHandler::Ar
sh::filesystemhandlers::LocalFilesystemHandler:(C++ class), 328, 860, 1466
(C++ function), 324, 326, 857, 858, 1462,
(C++ class), 328, 860, 1466
1464
sh::filesystemhandlers::SharcFilesystemHandler::Ar
sh::filesystemhandlers::LocalFilesystemHandler:(C++ member), 332
(C++ function), 324, 326, 856, 858, 1462,
sh::filesystemhandlers::SharcFilesystemHandler::Ar
1464
(C++ function), 333
sh::filesystemhandlers::LocalFilesystemHandler::Ar
sh::filesystemhandlers::LocalFilesystemHandler:(C++ function), 333
(C++ function), 324, 326, 856, 858, 1462,
(C++ function), 333
1463
sh::filesystemhandlers::SharcFilesystemHandler::Ar
sh::filesystemhandlers::LocalFilesystemHandler:(C++ function), 332
(C++ function), 324, 326, 856, 858, 1462,
sh::filesystemhandlers::SharcFilesystemHandler::Ar
1463
(C++ function), 333
sh::filesystemhandlers::LocalFilesystemHandler::Ar
sh::filesystemhandlers::LocalFilesystemHandler:(C++ function), 333
(C++ function), 324, 326, 856, 858, 1462,
sh::filesystemhandlers::SharcFilesystemHandler::Ar
1463
(C++ function), 333
sh::filesystemhandlers::LocalFilesystemHandler::Ar
sh::filesystemhandlers::LocalFilesystemHandler:(C++ member), 333
(C++ function), 324, 325, 856, 858, 1462,
sh::filesystemhandlers::SharcFilesystemHandler::Ar
1463
(C++ member), 333
sh::filesystemhandlers::LocalFilesystemHandler::Ar
sh::filesystemhandlers::LocalFilesystemHandler:(C++ function), 332
(C++ function), 325, 326, 857, 858, 1463,
(C++ function), 332
1464
sh::filesystemhandlers::SharcFilesystemHandler::Ar
sh::filesystemhandlers::LocalFilesystemHandler:(C++ function), 332
(C++ function), 332

```



```

sh::filesystemhandlers::SharcfFilesystemHahndler::ArchivedSizeDetailColumn::isValueAvailable
(C++ function), 333 (C++ function), 331, 863, 1469
sh::filesystemhandlers::SharcfFilesystemHahndler::ArchivedSizeDetailColumn::isAligned
(C++ function), 333 (C++ function), 329, 331, 861, 863, 1467,
sh::filesystemhandlers::SharcfFilesystemHandler::ArchivedSizeDetailColumn::isValueAvailable
(C++ function), 332 sh::filesystemhandlers::SharcfFilesystemHandler::do
sh::filesystemhandlers::SharcfFilesystemHandler::ArchivedSizeDetailColumn::isVisible
(C++ function), 332 (C++ function), 329, 864, 1469
sh::filesystemhandlers::SharcfFilesystemHandler::ArchivedSizeDetailColumn::name
(C++ function), 332 (C++ function), 329, 331, 861, 864, 1467,
sh::filesystemhandlers::SharcfFilesystemHahndler::ArchivedSizeDetailColumn::name
(C++ function), 333 (C++ function), 330, 862, 1468
sh::filesystemhandlers::SharcfFilesystemHahndler::ArchivedSizeDetailColumn::name
(C++ function), 332 (C++ function), 330, 862, 1468
sh::filesystemhandlers::SharcfFilesystemHahndler::ArchivedSizeDetailColumn::name
(C++ function), 332 (C++ function), 330, 862, 1468
sh::filesystemhandlers::SharcfFilesystemHahndler::ArchivedSizeDetailColumn::name
(C++ function), 333 (C++ function), 328, 330, 861, 863, 1466,
sh::filesystemhandlers::SharcfFilesystemHandler::ArchivedSizeDetailColumn::value
(C++ function), 332 sh::filesystemhandlers::SharcfFilesystemHandler::get
sh::filesystemhandlers::SharcfFilesystemHandler::ArchivedSizeDetailColumn::value
(C++ function), 333 (C++ function), 328, 329, 861, 862, 1466,
sh::filesystemhandlers::SharcfFilesystemHahndler::ArchivedSizeDetailColumn::value
(C++ function), 329, 330, 861, 863, 1466, (C++ function), 328, 329, 861, 862, 1466,
sh::filesystemhandlers::SharcfFilesystemHahndler::ArchivedSizeDetailColumn::value
(C++ function), 329, 330, 861, 863, 1467, (C++ function), 328, 329, 861, 862, 1466,
sh::filesystemhandlers::SharcfFilesystemHahndler::ArchivedSizeDetailColumn::value
(C++ function), 329, 331, 861, 863, 1467, (C++ function), 328, 329, 861, 862, 1466,
sh::filesystemhandlers::SharcfFilesystemHahndler::ArchivedSizeDetailColumn::value
(C++ function), 328, 330, 861, 863, 1466, (C++ function), 332, 864, 1470
sh::filesystemhandlers::SharcfFilesystemHandler::ArchivedSizeDetailColumn::value
(C++ function), 329, 331, 861, 863, 1467, sh::filesystemhandlers::SharcfFilesystemHandler::is
sh::filesystemhandlers::SharcfFilesystemHandler::ArchivedSizeDetailColumn::value
(C++ function), 329, 331, 861, 863, 1467, sh::filesystemhandlers::SharcfFilesystemHandler::ite
sh::filesystemhandlers::SharcfFilesystemHandler::ArchivedSizeDetailColumn::value
(C++ function), 328, 329, 861, 862, 1466,
sh::filesystemhandlers::SharcfFilesystemHandler::ArchivedSizeDetailColumn::value
(C++ function), 328, 329, 861, 862, 1466, sh::filesystemhandlers::SharcfFilesystemHandler::lis
sh::filesystemhandlers::SharcfFilesystemHahndler::ArchivedSizeDetailColumn::value
(C++ function), 329, 330, 861, 863, 1466, (C++ function), 332, 865, 1470
sh::filesystemhandlers::SharcfFilesystemHahndler::ArchivedSizeDetailColumn::value
(C++ function), 329, 331, 861, 863, 1467, sh::filesystemhandlers::SharcfFilesystemHandler::loc
sh::filesystemhandlers::SharcfFilesystemHandler::ArchivedSizeDetailColumn::value
(C++ function), 329, 331, 861, 863, 1467, sh::filesystemhandlers::SharcfFilesystemHandler::mod
sh::filesystemhandlers::SharcfFilesystemHandler::ArchivedSizeDetailColumn::value
(C++ function), 329, 331, 861, 863, 1467, sh::filesystemhandlers::SharcfFilesystemHandler::QB
sh::filesystemhandlers::SharcfFilesystemHahndler::ArchivedSizeDetailColumn::value
(C++ function), 329, 330, 861, 863, 1467, (C++ member), 334
sh::filesystemhandlers::SharcfFilesystemHahndler::ArchivedSizeDetailColumn::value
(C++ function), 329, 331, 861, 863, 1467, sh::filesystemhandlers::SharcfFilesystemHandler::QB
sh::filesystemhandlers::SharcfFilesystemHandler::ArchivedSizeDetailColumn::value
(C++ function), 331, 864, 1469 sh::filesystemhandlers::SharcfFilesystemHandler::QB

```

(C++ function), 334  
sh::filesystemhandlers::SharcfFileSystemHahdlérFileSystemHandler: (C++ function), 338, 869, 1474  
(C++ function), 330, 862, 1468  
(C++ function), 337, 868, 1473  
sh::filesystemhandlers::SharcfFileSystemHahdlérFileSystemHandler: (C++ function), 329, 331, 861, 864, 1467, 1469  
(C++ function), 336, 867, 1472  
sh::filesystemhandlers::SharcfFileSystemHandler: (C++ function), 331, 864, 1469  
sh::filesystemhandlers::SharcfFileSystemHandler: (C++ function), 336, 867, 1472, 1473  
(C++ function), 331, 864, 1469  
sh::filesystemhandlers::SharcfFileSystemHandler: (C++ function), 336, 867, 1472  
(C++ function), 330, 863, 1468  
sh::filesystemhandlers::SharcfFileSystemHandler: (C++ function), 335, 865, 866, 1471  
(C++ function), 330, 862, 1468  
sh::filesystemhandlers::SharcfFileSystemHandler: (C++ function), 335, 865, 866, 1471  
(C++ function), 328, 329, 861, 862, 1466, 1467  
sh::filesystemhandlers::SharcfFileSystemHahdlérFileSystemHandler: (C++ function), 332, 865, 1470  
sh::filesystemhandlers::SharcfFileSystemHahdlérFileSystemHandler: (C++ function), 332, 864, 1470  
sh::filesystemhandlers::SharcfFileSystemHahdlérFileSystemHandler: (C++ function), 331, 864, 1469  
sh::filesystemhandlers::SharcfFileSystemHahdlérFileSystemHandler: (C++ function), 331, 864, 1469  
sh::filesystemhandlers::SharcfFileSystemHahdlérFileSystemHandler: (C++ function), 331, 864, 1469  
sh::filesystemhandlers::SharcfFileSystemHahdlérFileSystemHandler: (C++ function), 338, 869, 1474  
sh::filesystemhandlers::WindowsNetworkFishesystemHandler: (C++ class), 334, 865, 1470  
sh::filesystemhandlers::WindowsNetworkFishesystemHandler: (C++ function), 337, 868, 1473  
sh::filesystemhandlers::WindowsNetworkFishesystemHandler: (C++ function), 335, 866, 1471  
sh::filesystemhandlers::WindowsNetworkFishesystemHandler: (C++ function), 334, 865, 1470  
sh::filesystemhandlers::WindowsNetworkFishesystemHandler: (C++ function), 335, 866, 1471  
sh::filesystemhandlers::WindowsNetworkFishesystemHandler: (C++ function), 336, 867, 1472  
sh::filesystemhandlers::WindowsNetworkFishesystemHandler: (C++ function), 335, 866, 1471  
sh::filesystemhandlers::WindowsNetworkFishesystemHandler: (C++ function), 337, 868, 1473  
sh::filesystemhandlers::WindowsNetworkFishesystemHandler: (C++ function), 336, 867, 1472  
sh::filesystemhandlers::WindowsNetworkFishesystemHandler: (C++ function), 338, 869, 1474  
sh::filesystemhandlers::WindowsNetworkFishesystemHandler: (C++ function), 335, 866, 1471  
sh::filesystemhandlers::WindowsNetworkFishesystemHandler: (C++ function), 337, 868, 1473  
sh::filesystemhandlers::WindowsNetworkFishesystemHandler: (C++ function), 336, 867, 1472  
sh::filesystemhandlers::WindowsNetworkFishesystemHandler: (C++ function), 336, 867, 1472  
sh::filesystemhandlers::WindowsNetworkFishesystemHandler: (C++ function), 334, 865, 1470  
sh::filesystemhandlers::WindowsNetworkFishesystemHandler: (C++ function), 337, 868, 1473  
sh::filesystemhandlers::WindowsNetworkFishesystemHandler: (C++ function), 335, 866, 1471, 1472  
sh::filesystemhandlers::WindowsNetworkFishesystemHandler: (C++ function), 337, 868, 1473  
sh::filesystemhandlers::WindowsNetworkFishesystemHandler: (C++ function), 335, 866, 1471  
sh::filesystemhandlers::WindowsNetworkFishesystemHandler: (C++ function), 336, 337, 868, 1473  
sh::filesystemhandlers::WindowsNetworkFishesystemHandler: (C++ function), 338, 869, 1474  
sh::filesystemhandlers::WindowsNetworkFishesystemHandler: (C++ function), 334, 865, 1470  
sh::filesystemhandlers::WindowsNetworkFishesystemHandler: (C++ function), 337, 868, 1473  
sh::filesystemhandlers::WindowsNetworkFishesystemHandler: (C++ function), 338, 869, 1474  
sh::filesystemhandlers::WindowsNetworkFishesystemHandler: (C++ function), 336, 867, 1472  
sh::filesystemhandlers::WindowsNetworkFishesystemHandler: (C++ function), 337, 868, 1473  
sh::filesystemhandlers::WindowsNetworkFishesystemHandler: (C++ function), 336, 867, 1472

(C++ function), 337, 868, 1473  
 sh::MainInit (C++ class), 21, 598  
 sh::MainInit::caughtSigterm (C++ member), 21, 598  
 sh::MainInit::doInitialize (C++ function), 21, 598  
 sh::MainInit::doShutdown (C++ function), 21, 598  
 sh::MainInit::singletonInitializer (C++ member), 21, 598  
 sh::paneldetails (C++ type), 869, 1474  
 sh::paneldetails::CommonPanelDetails (C++ class), 338, 869, 1474  
 sh::paneldetails::CommonPanelDetails::\_VshPanelDetails (C++ function), 338, 869, 1475  
 sh::paneldetails::CommonPanelDetails::doHitpanelDetails (C++ function), 338, 869, 1475  
 sh::paneldetails::CommonPanelDetails::doShutdown (C++ function), 338, 869, 1475  
 sh::paneldetails::CommonPanelDetails::FishModifiedTime (C++ function), 338, 870, 1475  
 sh::paneldetails::CommonPanelDetails::FishSize (C++ function), 338, 870, 1475  
 sh::paneldetails::CommonPanelDetails::FishType (C++ function), 338, 870, 1475  
 sh::paneldetails::CommonPanelDetails::NumberOfSelectedItems (C++ function), 338, 869, 1475  
 sh::paneldetails::CommonPanelDetails::TotalFiles (C++ function), 338, 870, 1475  
 sh::paneldetails::PanelDetail (C++ class), 339, 870, 1475  
 sh::paneldetails::PanelDetail::\_emit\_linkTriggered (C++ function), 339, 870, 1475  
 sh::paneldetails::PanelDetail::\_positionIndex (C++ member), 339, 871, 1476  
 sh::paneldetails::PanelDetail::\_rows (C++ member), 339, 871, 1476  
 sh::paneldetails::PanelDetail::\_rowsmutex (C++ member), 339, 871, 1476  
 sh::paneldetails::PanelDetail::\_valueWidthHint (C++ member), 339, 871, 1476  
 sh::paneldetails::PanelDetail::addOrReplaceRow (C++ function), 339, 870, 1475  
 sh::paneldetails::PanelDetail::addRow (C++ function), 339, 870, 1475  
 sh::paneldetails::PanelDetail::changed (C++ function), 339, 870, 1476  
 sh::paneldetails::PanelDetail::linkTriggered (C++ function), 339, 870, 1476  
 sh::paneldetails::PanelDetail::PanelDetails (C++ function), 339, 870, 1475  
 sh::paneldetails::PanelDetail::positionIndex (C++ function), 339, 870, 1475  
 sh::paneldetails::PanelDetail::rows (C++ member), 339, 871, 1476  
 sh::paneldetails::PanelDetail::setRows (C++ function), 339, 870, 1475  
 sh::paneldetails::PanelDetail::valueWidthHint (C++ function), 339, 870, 1475  
 sh::paneldetails::PanelDetailFactory (C++ class), 340, 871, 1476  
 sh::paneldetails::PanelDetailFactory::construct (C++ function), 340, 871, 1476  
 sh::paneldetails::PanelDetailFactoryForFunctionMulti (C++ class), 340, 871, 1476  
 sh::paneldetails::PanelDetailFactoryForFunctionMulti (C++ function), 340, 871, 1476  
 sh::paneldetails::PanelDetailFactoryForFunctionMulti (C++ function), 340, 871, 1476  
 sh::paneldetails::PanelDetailFactoryForFunctionSingle (C++ class), 340, 871, 1476  
 sh::paneldetails::PanelDetailFactoryForFunctionSingle (C++ function), 340, 871, 1477  
 sh::paneldetails::PanelDetailFactoryForFunctionSingle (C++ function), 340, 871, 1477  
 sh::paneldetails::PanelDetailManager (C++ class), 341, 871, 1477  
 sh::paneldetails::PanelDetailManager::\_factoriesmulti (C++ member), 342, 872, 1478  
 sh::paneldetails::PanelDetailManager::\_factoriessingle (C++ member), 342, 872, 1478  
 sh::paneldetails::PanelDetailManager::\_mutex (C++ member), 342, 872, 1478  
 sh::paneldetails::PanelDetailManager::doInitialize (C++ function), 341, 872, 1477  
 sh::paneldetails::PanelDetailManager::doShutdown (C++ function), 341, 872, 1477  
 sh::paneldetails::PanelDetailManager::getDetails (C++ function), 341, 872, 1477  
 sh::paneldetails::PanelDetailManager::isAlive (C++ function), 341, 872, 1477  
 sh::paneldetails::PanelDetailManager::PanelDetailManager (C++ function), 341, 872, 1478  
 sh::paneldetails::PanelDetailManager::registerFactory (C++ function), 341, 872, 1477  
 sh::paneldetails::PanelDetailManager::registerFactory (C++ function), 341, 872, 1477  
 sh::paneldetails::PanelDetailManager::shutdown (C++ function), 341, 872, 1477  
 sh::paneldetails::PanelDetailPositionIndex (C++ enum), 869, 1474  
 sh::paneldetails::PanelDetailPositionIndex::Exotic (C++ enumerator), 869, 1474  
 sh::paneldetails::PanelDetailPositionIndex::Interest (C++ enumerator), 869, 1474  
 sh::paneldetails::PanelDetailPositionIndex::VeryInterested (C++ enumerator), 869, 1474  
 sh::paneldetails::PanelDetailRow (C++ member), 341, 872, 1477

```

class), 342, 872, 1478
sh::panelDetails::PanelDetailRow::_key
    (C++ member), 342, 873, 1478
sh::panelDetails::PanelDetailRow::_value
    (C++ member), 342, 873, 1478
sh::panelDetails::PanelDetailRow::key
    (C++ function), 342, 873, 1478
sh::panelDetails::PanelDetailRow::PanelDetailRow()
    (C++ function), 342, 873, 1478
sh::panelDetails::PanelDetailRow::value
    (C++ function), 342, 873, 1478
sh::panelDetails::PanelDetailRowValueElement
    (C++ class), 342, 873, 1478
sh::panelDetails::PanelDetailRowValueElement::_GetIcon
    (C++ member), 343, 874, 1479
sh::panelDetails::PanelDetailRowValueElement::_GetValue
    (C++ member), 343, 874, 1479
sh::panelDetails::PanelDetailRowValueElement::_LinkClicked
    (C++ member), 343, 874, 1479
sh::panelDetails::PanelDetailRowValueElement::_LinkFunction
    (C++ member), 343, 874, 1479
sh::panelDetails::PanelDetailRowValueElement::_SetText
    (C++ member), 343, 874, 1479
sh::panelDetails::PanelDetailRowValueElement::icon
    (C++ function), 342, 873, 1478
sh::panelDetails::PanelDetailRowValueElement::id
    (C++ function), 342, 873, 1478
sh::panelDetails::PanelDetailRowValueElement::label
    (C++ function), 343, 873, 1479
sh::panelDetails::PanelDetailRowValueElement::linkClicked
    (C++ function), 343, 873, 1479
sh::panelDetails::PanelDetailRowValueElement::linkFunction
    (C++ function), 343, 873, 1479
sh::panelDetails::PanelDetailRowValueElement::linkLabel
    (C++ function), 343, 873, 1478
sh::panelDetails::PanelDetailRowValueElement::parentRowValueElement
    (C++ function), 342, 873, 1478
sh::panelDetails::PanelDetailRowValueElement::text
    (C++ function), 342, 873, 1478
sh::scripting (C++ type), 874, 1479
sh::scripting::Api (C++ class), 343, 874, 1479
sh::scripting::api (C++ type), 880, 1485, 1525
sh::scripting::Api::_classes (C++ member), 344, 874, 1480
sh::scripting::Api::_classesByNativeTypes
    (C++ member), 344, 874, 1480
sh::scripting::Api::_rootMembers (C++ member), 344, 874, 1480
sh::scripting::Api::_rootMembersClasses
    (C++ member), 344, 874, 1480
sh::scripting::Api::~~Api (C++ function), 343, 874, 1479
sh::scripting::Api::Api (C++ function), 343, 874, 1479
sh::scripting::api::ApiActionActionItem
    (C++ class), 352, 880, 1485, 1525
sh::scripting::api::ApiActionActionItem::addActionScript
    (C++ member), 354, 882, 1487, 1527
sh::scripting::api::ApiActionActionItem::_execute
    (C++ function), 352, 880, 1486, 1526
sh::scripting::api::ApiActionActionItem::_initialize
    (C++ member), 354, 882, 1487, 1527
sh::scripting::api::ApiActionActionItem::_initialized
    (C++ member), 354, 882, 1487, 1527
sh::scripting::api::ApiActionActionItem::_setParent
    (C++ function), 353, 881, 1487, 1527
sh::scripting::api::ApiActionActionItem::action
    (C++ function), 352, 880, 1486, 1526
sh::scripting::api::ApiActionActionItem::ApiActionItem
    (C++ function), 352, 880, 1486, 1526
sh::scripting::api::ApiActionActionItem::changed
    (C++ function), 354, 882, 1487, 1527
sh::scripting::api::ApiActionActionItem::defaultAction
    (C++ function), 353, 881, 1486, 1526
sh::scripting::api::ApiActionActionItem::enabled
    (C++ function), 353, 881, 1486, 1526
sh::scripting::api::ApiActionActionItem::execute
    (C++ function), 352, 880, 1486, 1526
sh::scripting::api::ApiActionActionItem::icon
    (C++ function), 353, 881, 1486, 1526
sh::scripting::api::ApiActionActionItem::initialize
    (C++ function), 352, 880, 1486, 1526
sh::scripting::api::ApiActionActionItem::initialized
    (C++ member), 353, 881, 1487, 1527
sh::scripting::api::ApiActionActionItem::initialization
    (C++ member), 353, 881, 1487, 1527
sh::scripting::api::ApiActionActionItem::isCheckable
    (C++ function), 353, 881, 1486, 1526
sh::scripting::api::ApiActionActionItem::isChecked
    (C++ function), 353, 881, 1486, 1526
sh::scripting::api::ApiActionActionItem::isInitializable
    (C++ function), 353, 881, 1487, 1527
sh::scripting::api::ApiActionActionItem::isInitializing
    (C++ function), 354, 881, 1487, 1527
sh::scripting::api::ApiActionActionItem::parentActionItem
    (C++ function), 353, 881, 1487, 1527
sh::scripting::api::ApiActionActionItem::setChecked
    (C++ function), 353, 881, 1486, 1526
sh::scripting::api::ApiActionActionItem::setEnabled
    (C++ function), 353, 881, 1486, 1526
sh::scripting::api::ApiActionActionItem::setIcon
    (C++ function), 353, 881, 1486, 1526
sh::scripting::api::ApiActionActionItem::setShortcut
    (C++ function), 353, 880, 1486, 1526
sh::scripting::api::ApiActionActionItem::setText
    (C++ function), 353, 881, 1486, 1526
sh::scripting::api::ApiActionActionItem::setVisible
    (C++ function), 353, 881, 1487, 1527
sh::scripting::api::ApiActionActionItem::shortcutName
    (C++ function), 352, 880, 1486, 1526
sh::scripting::api::ApiActionActionItem::shortcuts
    (C++ function), 353, 880, 1486, 1526

```



```

sh::scripting::api::ApiActionActionItem:sh::scripting::api::ApiDetailColumn::registerKnown
  (C++ function), 353, 881, 1486, 1526                                (C++ function), 356, 884, 1489, 1529
sh::scripting::api::ApiActionActionItem:sh::scripting::api::ApiDetailColumn::requestValue
  (C++ function), 353, 881, 1487, 1527                                (C++ function), 355, 883, 1488, 1528
sh::scripting::api::ApiActionFactory      sh::scripting::api::ApiDetailColumn::sort_doTypedi
  (C++ class), 354, 882, 1487, 1527                                (C++ function), 355, 883, 1488, 1528
sh::scripting::api::ApiActionFactory::actionAvailable:sh::scripting::api::ApiDetailColumn::sort_order
  (C++ function), 354, 882, 1487, 1527                                (C++ function), 355, 883, 1488, 1528
sh::scripting::api::ApiActionFactory::ApiActionFactory:sh::scripting::api::ApiDetailColumn::value
  (C++ function), 354, 882, 1487, 1527                                (C++ function), 355, 882, 1488, 1528
sh::scripting::api::ApiActionFactory::construct:sh::scripting::api::ApiDetailColumn::VALUE_UNAVAILA
  (C++ function), 354, 882, 1487, 1527                                (C++ function), 356, 884, 1489, 1529
sh::scripting::api::ApiDetailColumn      sh::scripting::api::ApiFilePropertyDialogTab
  (C++ class), 354, 882, 1488, 1528                                (C++ class), 356, 884, 1489, 1529
sh::scripting::api::ApiDetailColumn::_applyValue:sh::scripting::api::ApiFilePropertyDialogTab::_butt
  (C++ member), 356, 884, 1489, 1529                                (C++ member), 357, 885, 1490, 1530
sh::scripting::api::ApiDetailColumn::_destroyValue:sh::scripting::api::ApiFilePropertyDialogTab::_prop
  (C++ member), 356, 884, 1489, 1529                                (C++ member), 357, 885, 1491, 1531
sh::scripting::api::ApiDetailColumn::ApiDetailColumn:sh::scripting::api::ApiFilePropertyDialogTab::_titl
  (C++ function), 354, 882, 1488, 1528                                (C++ member), 357, 885, 1491, 1531
sh::scripting::api::ApiDetailColumn::applyValue:sh::scripting::api::ApiFilePropertyDialogTab::ApiF
  (C++ function), 354, 882, 1488, 1528                                (C++ function), 356, 884, 1490, 1530
sh::scripting::api::ApiDetailColumn::applyValueByEnum:sh::scripting::api::ApiFilePropertyDialogTab::dialo
  (C++ function), 355, 883, 1489, 1528                                (C++ function), 357, 885, 1490, 1530
sh::scripting::api::ApiDetailColumn::applyValueByNode:sh::scripting::api::ApiFilePropertyDialogTab::nodes
  (C++ function), 355, 883, 1489, 1529                                (C++ function), 357, 885, 1490, 1530
sh::scripting::api::ApiDetailColumn::completeValue:sh::scripting::api::ApiFilePropertyDialogTab::popu
  (C++ function), 355, 883, 1488, 1528                                (C++ function), 356, 884, 1490, 1530
sh::scripting::api::ApiDetailColumn::defaultWidth:sh::scripting::api::ApiFilePropertyDialogTab::prop
  (C++ function), 355, 883, 1488, 1528                                (C++ function), 356, 884, 1490, 1530
sh::scripting::api::ApiDetailColumn::determineValue:sh::scripting::api::ApiFilePropertyDialogTab::PROPR
  (C++ function), 354, 882, 1488, 1528                                (C++ member), 357, 885, 1491, 1531
sh::scripting::api::ApiDetailColumn::displayIndex:sh::scripting::api::ApiFilePropertyDialogTab::PROPR
  (C++ function), 355, 883, 1488, 1528                                (C++ member), 357, 885, 1491, 1531
sh::scripting::api::ApiDetailColumn::DISPLAYINDEX_CORE:sh::scripting::api::ApiFilePropertyDialogTab::PROPR
  (C++ member), 356, 884, 1489, 1529                                (C++ member), 357, 885, 1491, 1531
sh::scripting::api::ApiDetailColumn::DISPLAYINDEX_EXOTIC:sh::scripting::api::ApiFilePropertyDialogTab::Prope
  (C++ member), 356, 884, 1489, 1529                                (C++ class), 357, 359, 885, 1491, 1531
sh::scripting::api::ApiDetailColumn::DISPLAYINDEX_INTERESTING:sh::scripting::api::ApiFilePropertyDialogTab::Prope
  (C++ member), 356, 884, 1489, 1529                                (C++ type), 357, 359, 885, 1491, 1531
sh::scripting::api::ApiDetailColumn::displayName:sh::scripting::api::ApiFilePropertyDialogTab::Prope
  (C++ function), 354, 882, 1488, 1528                                (C++ member), 358, 359, 886, 1491, 1531
sh::scripting::api::ApiDetailColumn::displayValue:sh::scripting::api::ApiFilePropertyDialogTab::Prope
  (C++ function), 355, 882, 1488, 1528                                (C++ function), 358, 359, 886, 1491, 1531
sh::scripting::api::ApiDetailColumn::findKnownDepainColumn:sh::scripting::api::ApiFilePropertyDialogTab::Prope
  (C++ function), 356, 884, 1489, 1529                                (C++ member), 358, 359, 886, 1491, 1531
sh::scripting::api::ApiDetailColumn::isRightAligned:sh::scripting::api::ApiFilePropertyDialogTab::Prope
  (C++ function), 355, 883, 1489, 1528                                (C++ member), 358, 359, 886, 1491, 1531
sh::scripting::api::ApiDetailColumn::isValueAvailable:sh::scripting::api::ApiFilePropertyDialogTab::refre
  (C++ function), 354, 882, 1488, 1528                                (C++ function), 356, 884, 1490, 1530
sh::scripting::api::ApiDetailColumn::isVisible:sh::scripting::api::ApiFilePropertyDialogTab::Strin
  (C++ function), 355, 883, 1488, 1528                                (C++ type), 357, 885, 1491, 1531
sh::scripting::api::ApiDetailColumn::name:sh::scripting::api::ApiFilePropertyDialogTab::title
  (C++ function), 354, 882, 1488, 1528                                (C++ function), 356, 884, 1490, 1530

```

sh::scripting::api::ApiFilePropertyDialogTabScriptingWithApiMnApiFilesystemHandler::\_getMimeType  
 (C++ function), 356, 884, 1490, 1530 (C++ member), 363, 891, 1497, 1537  
 sh::scripting::api::ApiFilePropertyDialogTabScriptingWidget::ApiFilesystemHandler::\_getMimeType  
 (C++ function), 356, 884, 1490, 1530 (C++ member), 363, 891, 1496, 1536  
 sh::scripting::api::ApiFilePropertyDialogTabScriptingWidget::ApiFilesystemHandler::\_getSize  
 (C++ function), 357, 885, 1490, 1530 (C++ member), 363, 891, 1496, 1536  
 sh::scripting::api::ApiFilePropertyDialogTabScriptingWidget::ApiFilesystemHandler::\_getType  
 (C++ function), 357, 885, 1490, 1530 (C++ member), 363, 891, 1496, 1536  
 sh::scripting::api::ApiFilePropertyDialogTabScripting::api::ApiFilesystemHandler::\_itemlist  
 (C++ class), 358, 886, 1491, 1531 (C++ member), 364, 892, 1497, 1537  
 sh::scripting::api::ApiFilePropertyDialogTabScripting::ApiFilePropertyDialogTabFactory::removeE  
 (C++ function), 358, 886, 1491, 1531 (C++ member), 363, 891, 1497, 1537  
 sh::scripting::api::ApiFilePropertyDialogTabScripting::api::ApiFilesystemHandler::\_renameIt  
 (C++ function), 358, 886, 1491, 1531 (C++ member), 364, 892, 1497, 1537  
 sh::scripting::api::ApiFilePropertyDialogTabScripting::REGISTEROFFILEPROPERTIESMANALOGTABSENEXT  
 (C++ member), 358, 886, 1492, 1532 (C++ member), 363, 891, 1497, 1537  
 sh::scripting::api::ApiFilePropertyDialogTabScripting::REGISTEROFFILEPROPERTIESMANALOGTABSENEXX  
 (C++ member), 358, 886, 1492, 1532 (C++ member), 363, 891, 1496, 1536  
 sh::scripting::api::ApiFilePropertyDialogTabScripting::REGISTEROFFILEPROPERTIESMANALOGTABSENEXM  
 (C++ member), 358, 886, 1492, 1532 (C++ member), 363, 891, 1496, 1536  
 sh::scripting::api::ApiFilePropertyDialogTabScripting::REGISTEROFFILEPROPERTIESMANALOGTABPINDEXS  
 (C++ member), 358, 886, 1492, 1532 (C++ function), 359, 887, 1492, 1532  
 sh::scripting::api::ApiFilePropertyDialogTabScripting::REGISTEROFFILEPROPERTIESMANALOGTABPINDEXT  
 (C++ member), 358, 886, 1492, 1532 (C++ function), 360, 361, 887, 889, 1493,  
 sh::scripting::api::ApiFilesystemHandler 1495, 1533, 1535  
 (C++ class), 359, 886, 1492, 1532 sh::scripting::api::ApiFilesystemHandler::canCreate  
 sh::scripting::api::ApiFilesystemHandler::\_canCreate (C++ function), 360, 362, 888, 889, 1493,  
 (C++ member), 363, 891, 1497, 1537 1495, 1533, 1535  
 sh::scripting::api::ApiFilesystemHandler::sh::canCreateFile (C++ function), 360, 362, 888, 889, 1493,  
 (C++ member), 363, 891, 1497, 1537 1495, 1533, 1535  
 sh::scripting::api::ApiFilesystemHandler::\_canDelete (C++ function), 360, 362, 888, 889, 1493,  
 (C++ member), 363, 891, 1497, 1537 1495, 1533, 1535  
 sh::scripting::api::ApiFilesystemHandler::sh::canDeleteFile (C++ function), 360, 362, 888, 889, 1493,  
 (C++ member), 364, 892, 1497, 1537 1495, 1533, 1535  
 sh::scripting::api::ApiFilesystemHandler::sh::canGetFileContent (C++ function), 360, 361, 887, 889, 1493,  
 (C++ member), 364, 892, 1497, 1537 1495, 1533, 1535  
 sh::scripting::api::ApiFilesystemHandler::\_canRename (C++ function), 360, 362, 888, 889, 1493,  
 (C++ member), 364, 891, 1497, 1537 1495, 1533, 1535  
 sh::scripting::api::ApiFilesystemHandler::sh::canRenameFile (C++ function), 360, 362, 888, 889, 1493,  
 (C++ member), 364, 892, 1497, 1537 1495, 1533, 1535  
 sh::scripting::api::ApiFilesystemHandler::sh::createDirectory (C++ function), 359, 360, 887, 888, 1492,  
 (C++ member), 363, 891, 1497, 1537 1495, 1532, 1534  
 sh::scripting::api::ApiFilesystemHandler::\_createDir (C++ function), 360, 361, 887, 889, 1493,  
 (C++ member), 364, 892, 1497, 1537 1495, 1533, 1535  
 sh::scripting::api::ApiFilesystemHandler::sh::createFile (C++ function), 360, 362, 888, 889, 1493,  
 (C++ member), 364, 891, 1497, 1537 1495, 1533, 1535  
 sh::scripting::api::ApiFilesystemHandler::sh::deleteFile (C++ function), 360, 362, 888, 889, 1493,  
 (C++ member), 364, 891, 1497, 1537 1495, 1533, 1535  
 sh::scripting::api::ApiFilesystemHandler::\_getExtendedAttribute (C++ member), 363, 891, 1496, 1536  
 sh::scripting::api::ApiFilesystemHandler::sh::createLink (C++ function), 360, 362, 888, 889, 1493,  
 (C++ member), 363, 891, 1496, 1536 1495, 1533, 1535  
 sh::scripting::api::ApiFilesystemHandler::sh::getLinkTarget (C++ function), 363, 890, 1496, 1536

sh::scripting::api::ApiFilesystemHandler::delete(C++ function), 362, 890, 1496, 1536  
 (C++ function), 362, 890, 1495, 1535 sh::scripting::api::ApiFilesystemHandler::setCustom  
 sh::scripting::api::ApiFilesystemHandler::delete(C++ function), 360, 361, 887, 889, 1493,  
 (C++ function), 360, 362, 888, 890, 1493, 1495, 1533, 1534  
 1495, 1533, 1535 sh::scripting::api::ApiFilesystemHandler::setExtens  
 sh::scripting::api::ApiFilesystemHandler::getAd(C++ function), 360, 361, 887, 889, 1493,  
 (C++ function), 360, 362, 888, 890, 1493, 1494, 1533, 1534  
 1496, 1533, 1535 sh::scripting::api::ApiFilesystemHandler::setMtime  
 sh::scripting::api::ApiFilesystemHandler::getCu(C++ function), 359, 361, 887, 888, 1492,  
 (C++ function), 360, 361, 887, 889, 1493, 1494, 1532, 1534  
 1494, 1533, 1534 sh::scripting::api::ApiFilesystemHandler::viewEnter  
 sh::scripting::api::ApiFilesystemHandler::getEx(C++ function), 362, 890, 1496, 1536  
 (C++ function), 359, 361, 887, 889, 1493, sh::scripting::api::ApiFilesystemHandler::viewLeft  
 1494, 1533, 1534 (C++ function), 362, 890, 1496, 1536  
 sh::scripting::api::ApiFilesystemHandler::getEx(C++ function), 362, 890, 1496, 1536  
 (C++ function), 359, 361, 887, 889, 1493, (C++ class), 364, 892, 1497, 1537  
 1494, 1533, 1534 sh::scripting::api::ApiFilesystemOperationProgressSh  
 sh::scripting::api::ApiFilesystemHandler::getFi(C++ member), 365, 893, 1498, 1538  
 (C++ function), 360, 361, 887, 889, 1493, sh::scripting::api::ApiFilesystemOperationProgressSh  
 1495, 1533, 1535 (C++ member), 365, 893, 1498, 1538  
 sh::scripting::api::ApiFilesystemHandler::getLinkTarget(C++ function), 364, 892, 1498, 1538  
 (C++ function), 359, 361, 887, 888, 1492, (C++ function), 364, 892, 1498, 1538  
 1494, 1532, 1534 sh::scripting::api::ApiFilesystemOperationProgressSh  
 sh::scripting::api::ApiFilesystemHandler::getMi(C++ function), 364, 892, 1498, 1538  
 (C++ function), 359, 361, 887, 888, 1492, sh::scripting::api::ApiFilesystemOperationProgressSh  
 1494, 1532, 1534 (C++ function), 364, 892, 1498, 1538  
 sh::scripting::api::ApiFilesystemHandler::getM(C++ function), 364, 892, 1498, 1538  
 (C++ function), 359, 361, 887, 888, 1492, (C++ function), 364, 892, 1498, 1538  
 1494, 1532, 1534 sh::scripting::api::ApiFilesystemOperationProgressSh  
 sh::scripting::api::ApiFilesystemHandler::getSi(C++ function), 364, 892, 1498, 1538  
 (C++ function), 359, 360, 887, 888, 1492, sh::scripting::api::ApiFilesystemOperationProgressSh  
 1494, 1532, 1534 (C++ function), 365, 892, 1498, 1538  
 sh::scripting::api::ApiFilesystemHandler::getSt(C++ function), 364, 892, 1498, 1538  
 (C++ function), 359, 360, 887, 888, 1492, (C++ function), 364, 892, 1498, 1538  
 1494, 1532, 1534 sh::scripting::api::ApiFilesystemOperationProgressSh  
 sh::scripting::api::ApiFilesystemHandler::isWel(C++ function), 364, 892, 1498, 1538  
 (C++ function), 362, 890, 1496, 1536 sh::scripting::api::ApiFilesystemOperationProgressSh  
 sh::scripting::api::ApiFilesystemHandler::iteml(C++ function), 364, 892, 1498, 1538  
 (C++ function), 359, 360, 887, 888, 1492, sh::scripting::api::ApiFilesystemOperationProgressSh  
 1493, 1532, 1533 (C++ function), 364, 892, 1498, 1538  
 sh::scripting::api::ApiFilesystemHandler::sh(C++ function), 364, 892, 1498, 1538  
 (C++ function), 359, 361, 887, 888, 1492, (C++ class), 365, 893, 1498, 1538  
 1494, 1532, 1534 sh::scripting::api::ApiGlobalObject::\_createFilesys  
 sh::scripting::api::ApiFilesystemHandler::model(C++ function), 368, 896, 1502, 1542  
 (C++ function), 363, 891, 1496, 1536 sh::scripting::api::ApiGlobalObject::\_createReadDat  
 sh::scripting::api::ApiFilesystemHandler::remov(C++ function), 369, 897, 1503, 1543  
 (C++ function), 360, 361, 887, 889, 1493, sh::scripting::api::ApiGlobalObject::\_findFilesystem  
 1494, 1533, 1534 (C++ function), 368, 897, 1502, 1542  
 sh::scripting::api::ApiFilesystemHandler::sh(C++ function), 364, 892, 1498, 1538  
 (C++ function), 360, 362, 888, 890, 1493, (C++ function), 368, 896, 1502, 1542  
 1495, 1533, 1535 sh::scripting::api::ApiGlobalObject::\_register\_File  
 sh::scripting::api::ApiFilesystemHandler::requi(C++ function), 368, 897, 1502, 1542  
 (C++ function), 362, 890, 1496, 1536 sh::scripting::api::ApiGlobalObject::\_tryGetFilesys  
 sh::scripting::api::ApiFilesystemHandler::search(C++ function), 368, 897, 1503, 1542

sh::scripting::api::ApiGlobalObject::actshnddefcraupgcedpnceApiGlobalObject::specraabpsearch  
(C++ function), 365, 893, 1499, 1539 (C++ function), 369, 898, 1503, 1543

sh::scripting::api::ApiGlobalObject::actshnddefcraupgcedpnceApiGlobalObject::create\_Submenu  
(C++ function), 365, 893, 1499, 1539 (C++ function), 367, 895, 1500, 1540

sh::scripting::api::ApiGlobalObject::ApiGlobalObject::create\_Thread  
(C++ function), 371, 900, 1506, 1545 (C++ function), 369, 897, 1503, 1543

sh::scripting::api::ApiGlobalObject::bookmarkManager::create\_Thumbnail  
(C++ function), 371, 900, 1505, 1545 (C++ function), 368, 896, 1502, 1542

sh::scripting::api::ApiGlobalObject::configuraphcategory::ApiGlobalObject::create\_Timer  
(C++ function), 366, 893, 1499, 1539 (C++ function), 369, 897, 1503, 1543

sh::scripting::api::ApiGlobalObject::configuraphcategory::ApiGlobalObject::createArgument  
(C++ function), 366, 893, 1499, 1539 (C++ function), 368, 897, 1503, 1542

sh::scripting::api::ApiGlobalObject::configuraphcategory::ApiGlobalObject::createUrl  
(C++ function), 366, 893, 1499, 1539 (C++ function), 368, 897, 1503, 1542

sh::scripting::api::ApiGlobalObject::create\_AcrophAnticapiemApiGlobalObject::createException  
(C++ function), 367, 894, 1500, 1540 (C++ function), 369, 897, 1503, 1543

sh::scripting::api::ApiGlobalObject::create\_AcrophAnticapiemApiGlobalObject::createIOException  
(C++ function), 367, 895, 1501, 1541 (C++ function), 368, 897, 1503, 1542

sh::scripting::api::ApiGlobalObject::create\_ByRegExprPredictApiGlobalObject::createOperation  
(C++ function), 371, 900, 1505, 1545 (C++ function), 368, 897, 1503, 1542

sh::scripting::api::ApiGlobalObject::create\_BerapiCndumapi::ApiGlobalObject::createProgram  
(C++ function), 367, 895, 1501, 1541 (C++ function), 369, 897, 1503, 1542

sh::scripting::api::ApiGlobalObject::create\_BerapiFactapiMulApiGlobalObject::createRuntime  
(C++ function), 367, 896, 1501, 1541 (C++ function), 369, 897, 1503, 1542

sh::scripting::api::ApiGlobalObject::create\_BerapiFactapiSingleApiGlobalObject::createSetting  
(C++ function), 367, 896, 1501, 1541 (C++ function), 369, 897, 1503, 1543

sh::scripting::api::ApiGlobalObject::create\_DonResndvchpksPreGlobalObject::displayindex\_  
(C++ function), 371, 900, 1505, 1545 (C++ function), 365, 893, 1499, 1539

sh::scripting::api::ApiGlobalObject::create\_FchEPropertyDialogGlobalObject::displayindex\_  
(C++ function), 367, 895, 1501, 1541 (C++ function), 366, 893, 1499, 1539

sh::scripting::api::ApiGlobalObject::create\_FchEPropertyDialogGlobalObject::displayindex\_  
(C++ function), 367, 895, 1501, 1541 (C++ function), 365, 893, 1499, 1539

sh::scripting::api::ApiGlobalObject::create\_FchEsysngmHandleApiGlobalObject::doInitialize  
(C++ function), 367, 894, 1500, 1540 (C++ function), 371, 900, 1505, 1545

sh::scripting::api::ApiGlobalObject::create\_FchEsysngmOperatApiGlobalObject::doShutdown  
(C++ function), 368, 896, 1502, 1541 (C++ function), 371, 900, 1505, 1545

sh::scripting::api::ApiGlobalObject::create\_HchEOnCngreapDirApiGlobalObject::editcfilepropertyd  
(C++ function), 371, 900, 1505, 1545 (C++ function), 366, 894, 1499, 1539

sh::scripting::api::ApiGlobalObject::create\_HchEOnSedetchapOnLaveGProbdObject::filepropertyd  
(C++ function), 371, 900, 1505, 1545 (C++ function), 366, 894, 1500, 1540

sh::scripting::api::ApiGlobalObject::create\_Key\$pringutPreApiGlobalObject::filepropertyd  
(C++ function), 371, 900, 1505, 1545 (C++ function), 366, 894, 1500, 1540

sh::scripting::api::ApiGlobalObject::create\_OnDipchngriepPreApiGlobalObject::filepropertyd  
(C++ function), 371, 900, 1505, 1545 (C++ function), 366, 894, 1500, 1540

sh::scripting::api::ApiGlobalObject::create\_OnFipchngedapateApiGlobalObject::filepropertyd  
(C++ function), 371, 900, 1505, 1545 (C++ function), 366, 894, 1500, 1540

sh::scripting::api::ApiGlobalObject::create\_OnLipchngedapateApiGlobalObject::filepropertyd  
(C++ function), 371, 900, 1505, 1545 (C++ function), 366, 894, 1500, 1540

sh::scripting::api::ApiGlobalObject::create\_OnSipglantaySelApiGlobalObject::filepropertyd  
(C++ function), 371, 900, 1505, 1545 (C++ function), 366, 894, 1500, 1540

sh::scripting::api::ApiGlobalObject::create\_PosipionCndexPreApiGlobalObject::filepropertyd  
(C++ function), 371, 900, 1505, 1545 (C++ function), 366, 894, 1500, 1540

sh::scripting::api::ApiGlobalObject::create\_SearchingCngreapOnApiGlobalObject::filesystemnode  
(C++ function), 369, 897, 1503, 1543 (C++ function), 365, 893, 1498, 1538



sh::scripting::api::ApiGlobalObject::filesystemmptdengpeafileApiGlobalObject::operationstep\_ (C++ function), 365, 893, 1498, 1538 (C++ function), 366, 894, 1499, 1539  
 sh::scripting::api::ApiGlobalObject::filesystemmptdengpeafirstApiGlobalObject::operationstep\_ (C++ function), 365, 893, 1499, 1539 (C++ function), 366, 894, 1499, 1539  
 sh::scripting::api::ApiGlobalObject::filesystemmptdengpeapastApiGlobalObject::operationstep\_ (C++ function), 365, 893, 1499, 1539 (C++ function), 366, 894, 1499, 1539  
 sh::scripting::api::ApiGlobalObject::filesystemmptdengpeapastAppGlobalObject::operationstep\_ (C++ function), 365, 893, 1499, 1539 (C++ function), 366, 894, 1499, 1539  
 sh::scripting::api::ApiGlobalObject::filesystemmptdengpeapinkApiGlobalObject::operationstep\_ (C++ function), 365, 893, 1498, 1538 (C++ function), 366, 894, 1499, 1539  
 sh::scripting::api::ApiGlobalObject::filesystemmptdengpeaponeApiGlobalObject::paneldetail\_p (C++ function), 365, 893, 1499, 1539 (C++ function), 366, 894, 1499, 1539  
 sh::scripting::api::ApiGlobalObject::filesystemmptdengpeaspecApiGlobalObject::paneldetail\_p (C++ function), 365, 893, 1499, 1539 (C++ function), 366, 894, 1499, 1539  
 sh::scripting::api::ApiGlobalObject::filesystemmptdengpeapunkApiGlobalObject::paneldetail\_p (C++ function), 365, 893, 1498, 1538 (C++ function), 366, 894, 1499, 1539  
 sh::scripting::api::ApiGlobalObject::findDetailCptimgByNameApiGlobalObject::refreshData (C++ function), 370, 899, 1505, 1544 (C++ function), 368, 897, 1503, 1542  
 sh::scripting::api::ApiGlobalObject::getEhrlFromStringApiGlobalObject::register\_Conf (C++ function), 365, 893, 1498, 1538 (C++ function), 370, 899, 1504, 1544  
 sh::scripting::api::ApiGlobalObject::isAhvescripting::api::ApiGlobalObject::register\_Conf (C++ function), 371, 900, 1506, 1545 (C++ function), 369, 898, 1504, 1543  
 sh::scripting::api::ApiGlobalObject::isDebugBuildting::api::ApiGlobalObject::register\_Conf (C++ function), 367, 894, 1500, 1540 (C++ function), 369, 898, 1503, 1543  
 sh::scripting::api::ApiGlobalObject::logDebugscripting::api::ApiGlobalObject::register\_Conf (C++ function), 365, 893, 1498, 1538 (C++ function), 370, 899, 1504, 1544  
 sh::scripting::api::ApiGlobalObject::logEhroscripiting::api::ApiGlobalObject::register\_pred (C++ function), 365, 893, 1498, 1538 (C++ function), 366, 894, 1500, 1540  
 sh::scripting::api::ApiGlobalObject::logEhfoscripting::api::ApiGlobalObject::registerFileP (C++ function), 365, 893, 1498, 1538 (C++ function), 369, 898, 1503, 1543  
 sh::scripting::api::ApiGlobalObject::logWarningipting::api::ApiGlobalObject::registerPanelL (C++ function), 365, 893, 1498, 1538 (C++ function), 369, 898, 1503, 1543  
 sh::scripting::api::ApiGlobalObject::mainwindowipting::api::ApiGlobalObject::registerPanelL (C++ function), 369, 898, 1503, 1543 (C++ function), 369, 898, 1503, 1543  
 sh::scripting::api::ApiGlobalObject::messageboxipting::capitelApiGlobalObject::registerSearch (C++ function), 365, 893, 1499, 1539 (C++ function), 369, 898, 1503, 1543  
 sh::scripting::api::ApiGlobalObject::messageboxipting::capitineApiGlobalObject::registerSettin (C++ function), 365, 893, 1499, 1539 (C++ function), 369, 898, 1503, 1543  
 sh::scripting::api::ApiGlobalObject::messageboxipting::napi::ApiGlobalObject::registerThumbr (C++ function), 365, 893, 1499, 1539 (C++ function), 369, 898, 1503, 1543  
 sh::scripting::api::ApiGlobalObject::messageboxipting::okapi::ApiGlobalObject::registerTrans (C++ function), 365, 893, 1499, 1539 (C++ function), 370, 899, 1505, 1544  
 sh::scripting::api::ApiGlobalObject::messageboxipting::replyApiGlobalObject::searchcriterio (C++ function), 365, 893, 1499, 1539 (C++ function), 366, 894, 1500, 1540  
 sh::scripting::api::ApiGlobalObject::messageboxipting::yapi::ApiGlobalObject::searchcriterio (C++ function), 365, 893, 1499, 1539 (C++ function), 366, 894, 1500, 1540  
 sh::scripting::api::ApiGlobalObject::modelHostNodeing::api::ApiGlobalObject::searchcriterio (C++ function), 368, 897, 1502, 1542 (C++ function), 366, 894, 1500, 1540  
 sh::scripting::api::ApiGlobalObject::openItemscripiting::api::ApiGlobalObject::settinggroup\_k (C++ function), 369, 898, 1503, 1543 (C++ function), 366, 894, 1499, 1539  
 sh::scripting::api::ApiGlobalObject::opeshatisippingnfaipctrAspGtubahQbjdirectsettinggroup\_c (C++ function), 366, 894, 1499, 1539 (C++ function), 366, 894, 1499, 1539  
 sh::scripting::api::ApiGlobalObject::opeshatisippingnfaipctrAspGtubahQmgedirectonngroup\_ (C++ function), 366, 894, 1499, 1539 (C++ function), 366, 894, 1499, 1539

sh::scripting::api::ApiGlobalObject::setshnggcriptngbaapi::ApiHelperMethods::ApiSubMenuAct  
(C++ function), 366, 893, 1499, 1539 (C++ function), 373, 902, 1508, 1547

sh::scripting::api::ApiGlobalObject::setshnggcriptngg::api::ApiHelperMethods::Configuration  
(C++ function), 366, 894, 1499, 1539 (C++ function), 372, 901, 1506, 1546

sh::scripting::api::ApiGlobalObject::setshnggcriptnggciapi::ApiHelperMethods::Configuration  
(C++ function), 366, 894, 1499, 1539 (C++ function), 372, 901, 1506, 1546

sh::scripting::api::ApiGlobalObject::shashotBurlpinRugapDirApiHelperMethods::Configuration  
(C++ function), 370, 900, 1505, 1544 (C++ function), 372, 901, 1506, 1546

sh::scripting::api::ApiGlobalObject::shashotBarapi::ApiHelperMethods::Configuration  
(C++ function), 370, 899, 1505, 1544 (C++ function), 372, 901, 1506, 1546

sh::scripting::api::ApiGlobalObject::shashotSangpageg::api::ApiHelperMethods::FilePropertyY  
(C++ function), 371, 900, 1505, 1545 (C++ function), 374, 903, 1508, 1548

sh::scripting::api::ApiGlobalObject::shashotSystemPhggiaDirApiHelperMethods::FilePropertyY  
(C++ function), 371, 900, 1505, 1545 (C++ function), 374, 903, 1508, 1548

sh::scripting::api::ApiGlobalObject::shashotUserPilingnDapi::ApiHelperMethods::FilesystemNo  
(C++ function), 371, 900, 1505, 1545 (C++ function), 372, 901, 1506, 1546

sh::scripting::api::ApiGlobalObject::shutdowscripting::api::ApiHelperMethods::FilesystemNo  
(C++ function), 371, 900, 1505, 1545 (C++ function), 372, 901, 1506, 1546

sh::scripting::api::ApiGlobalObject::thumbascriptvdr:apdexApHelperMethods::FilesystemNo  
(C++ function), 366, 894, 1500, 1540 (C++ function), 372, 901, 1506, 1546

sh::scripting::api::ApiGlobalObject::thumbascriptvdr:apdexApHelperMethods::FilesystemNo  
(C++ function), 366, 894, 1500, 1540 (C++ function), 372, 901, 1506, 1546

sh::scripting::api::ApiGlobalObject::thumbascriptvdr:apdexApHelperMethods::FilesystemNo  
(C++ function), 366, 894, 1500, 1540 (C++ function), 372, 901, 1506, 1546

sh::scripting::api::ApiGlobalObject::thumbascriptvdr:apdexApHelperMethods::FilesystemNo  
(C++ function), 366, 894, 1500, 1540 (C++ function), 372, 901, 1506, 1545

sh::scripting::api::ApiHelperMethods sh::scripting::api::ApiHelperMethods::FilesystemOpe  
(C++ class), 371, 900, 1506, 1545 (C++ function), 372, 901, 1507, 1546

sh::scripting::api::ApiHelperMethods::ActionExscriptingUserFeedBakHelmerMathBdx::FilesystemOpe  
(C++ function), 374, 903, 1508, 1548 (C++ function), 372, 901, 1507, 1546

sh::scripting::api::ApiHelperMethods::ActionExscriptingUserFeedBakHelmerMathBdx::FilesystemOpe  
(C++ function), 374, 903, 1508, 1548 (C++ function), 372, 901, 1506, 1546

sh::scripting::api::ApiHelperMethods::ApsActionAptingItemienApiHelperMethods::FilesystemOpe  
(C++ function), 373, 902, 1507, 1547 (C++ function), 372, 901, 1506, 1546

sh::scripting::api::ApiHelperMethods::ApsActionAptingItemiseApiHelperMethods::FilesystemOpe  
(C++ function), 373, 902, 1507, 1547 (C++ function), 373, 902, 1507, 1546

sh::scripting::api::ApiHelperMethods::ApsActionAptingItemiseApiHelperMethods::FilesystemOpe  
(C++ function), 373, 902, 1507, 1547 (C++ function), 373, 902, 1507, 1546

sh::scripting::api::ApiHelperMethods::ApsActionAptingItemiviApiHelperMethods::FilesystemOpe  
(C++ function), 373, 902, 1507, 1547 (C++ function), 372, 902, 1507, 1546

sh::scripting::api::ApiHelperMethods::ApsFilePropertyDiapogTapiHelperMethods::FilesystemOpe  
(C++ function), 374, 903, 1508, 1547 (C++ function), 372, 902, 1507, 1546

sh::scripting::api::ApiHelperMethods::ApsReadDripDengceapeadApiHelperMethods::FilesystemOpe  
(C++ function), 372, 901, 1506, 1545 (C++ function), 373, 903, 1508, 1547

sh::scripting::api::ApiHelperMethods::ApsReadDripDengceapeadApiHelperMethods::FilesystemOpe  
(C++ function), 372, 901, 1506, 1545 (C++ function), 373, 902, 1508, 1547

sh::scripting::api::ApiHelperMethods::ApsSubMenuActingnIapm:apiHelperMethods::FilesystemOpe  
(C++ function), 373, 902, 1508, 1547 (C++ function), 373, 903, 1508, 1547

sh::scripting::api::ApiHelperMethods::ApsSubMenuActingnIapm:apiHelperMethods::FilesystemOpe  
(C++ function), 373, 902, 1507, 1547 (C++ function), 373, 902, 1508, 1547

sh::scripting::api::ApiHelperMethods::ApsSubMenuActingnIapm:apiHelperMethods::FilesystemOpe  
(C++ function), 373, 902, 1507, 1547 (C++ function), 374, 903, 1508, 1547

sh::scripting::api::ApiHelperMethods::ApsSubMenuActingnIapm:apiHelperMethods::FilesystemOpe  
(C++ function), 373, 902, 1508, 1547 (C++ function), 374, 903, 1508, 1547

sh::scripting::api::ApiHelperMethods::FishsystemOpenngtiapProcApReMdDataDegetEtenghdata  
 (C++ function), 374, 903, 1508, 1547 (C++ function), 375, 904, 1510, 1549  
 sh::scripting::api::ApiHelperMethods::FishsystemOpenngtiapProcApReMdDataDehasBytesended  
 (C++ function), 373, 903, 1508, 1547 (C++ member), 375, 905, 1510, 1549  
 sh::scripting::api::ApiHelperMethods::FishsystemOpenngtiapProcApReMdDataDehasOnhemInfo  
 (C++ function), 373, 902, 1508, 1547 (C++ class), 376, 905, 1510, 1549  
 sh::scripting::api::ApiHelperMethods::MashWindowptgenGurapentDApSearyNdeeterion::\_configure  
 (C++ function), 373, 902, 1507, 1547 (C++ member), 376, 905, 1510, 1550  
 sh::scripting::api::ApiHelperMethods::MashWindowptgenGurapentDApSearyNdeeterion::\_createEdit  
 (C++ function), 373, 902, 1507, 1547 (C++ function), 376, 905, 1511, 1550  
 sh::scripting::api::ApiHelperMethods::OperationSpengonconfictResSearchCriteria::\_editorUpd  
 (C++ function), 373, 902, 1508, 1547 (C++ function), 376, 905, 1511, 1550  
 sh::scripting::api::ApiHelperMethods::PashDetailptsetRowapi::ApiSearchCriteria::\_match  
 (C++ function), 373, 902, 1507, 1547 (C++ member), 376, 905, 1510, 1550  
 sh::scripting::api::ApiHelperMethods::QIdeviceipreadg::api::ApiSearchCriteria::ApiSearchC  
 (C++ function), 372, 901, 1506, 1545 (C++ function), 376, 905, 1510, 1549  
 sh::scripting::api::ApiHelperMethods::QIdeviceipreadgillapi::ApiSearchCriteria::deserializ  
 (C++ function), 372, 901, 1506, 1545 (C++ function), 376, 905, 1511, 1550  
 sh::scripting::api::ApiPanelDetailFactoryMultscripting::api::ApiSearchCriteria::factory  
 (C++ class), 374, 903, 1508, 1548 (C++ function), 376, 905, 1510, 1549  
 sh::scripting::api::ApiPanelDetailFactoryMultscripting::api::ApiSearchCriteria::getsearchsp  
 (C++ member), 374, 904, 1509, 1548 (C++ function), 376, 905, 1510, 1549  
 sh::scripting::api::ApiPanelDetailFactoryMultscripting::api::ApiSearchCriteria::match  
 (C++ member), 374, 904, 1509, 1548 (C++ function), 376, 905, 1510, 1549  
 sh::scripting::api::ApiPanelDetailFactoryMultscripting::api::ApiSearchCriteria::match2  
 (C++ function), 374, 903, 1509, 1548 (C++ function), 376, 905, 1510, 1549  
 sh::scripting::api::ApiPanelDetailFactoryMultscripting::api::ApiSearchCriteria::searchspec  
 (C++ function), 374, 903, 1509, 1548 (C++ member), 376, 905, 1510, 1550  
 sh::scripting::api::ApiPanelDetailFactoryMultscripting::api::ApiSearchCriteria::serialize  
 (C++ function), 374, 903, 1509, 1548 (C++ function), 376, 905, 1510, 1549  
 sh::scripting::api::ApiPanelDetailFactoryMultscripting::api::ApiSearchCriteria::valuedescri  
 (C++ function), 374, 903, 1509, 1548 (C++ function), 376, 905, 1510, 1549  
 sh::scripting::api::ApiPanelDetailFactorySingleipting::api::ApiSearchCriteriaFactory  
 (C++ class), 375, 904, 1509, 1548 (C++ class), 376, 905, 1511, 1550  
 sh::scripting::api::ApiPanelDetailFactorySingleipting::api::ApiSearchCriteriaFactory::\_get  
 (C++ member), 375, 904, 1509, 1549 (C++ member), 377, 906, 1511, 1551  
 sh::scripting::api::ApiPanelDetailFactorySingleipting::api::ApiSearchCriteriaFactory::\_isv  
 (C++ member), 375, 904, 1509, 1549 (C++ member), 377, 906, 1511, 1551  
 sh::scripting::api::ApiPanelDetailFactorySingleipting::api::ApiSearchCriteriaFactory::Api  
 (C++ function), 375, 904, 1509, 1548 (C++ function), 377, 906, 1511, 1550  
 sh::scripting::api::ApiPanelDetailFactorySingleipting::api::ApiSearchCriteriaFactory::cons  
 (C++ function), 375, 904, 1509, 1548 (C++ function), 377, 906, 1511, 1550  
 sh::scripting::api::ApiPanelDetailFactorySingleipting::api::ApiSearchCriteriaFactory::des  
 (C++ function), 375, 904, 1509, 1548 (C++ function), 377, 906, 1511, 1550  
 sh::scripting::api::ApiPanelDetailFactorySingleipting::api::ApiSearchCriteriaFactory::edit  
 (C++ function), 375, 904, 1509, 1548 (C++ function), 377, 906, 1511, 1550  
 sh::scripting::api::ApiReadDataDevice sh::scripting::api::ApiSearchCriteriaFactory::edit  
 (C++ class), 375, 904, 1509, 1549 (C++ function), 377, 906, 1511, 1550  
 sh::scripting::api::ApiReadDataDevice::\_shdescripting::api::ApiSearchCriteriaFactory::isV  
 (C++ function), 375, 904, 1510, 1549 (C++ function), 377, 906, 1511, 1550  
 sh::scripting::api::ApiReadDataDevice::\_getdataipting::api::ApiSearchCriteriaFactory::key  
 (C++ member), 375, 904, 1510, 1549 (C++ function), 377, 906, 1511, 1550  
 sh::scripting::api::ApiReadDataDevice::ApnReadDataDeviceapi::ApiSetting (C++  
 (C++ function), 375, 904, 1510, 1549 class), 377, 906, 1511, 1551

sh::scripting::api::ApiSetting::\_description (C++ member), 378, 907, 1512, 1551  
 sh::scripting::api::ApiSetting::\_getValuesh::scripting::api::ApiSubmenuItem::enabled (C++ function), 379, 908, 1513, 1552  
 sh::scripting::api::ApiSetting::\_group sh::scripting::api::ApiSubmenuItem::icon (C++ member), 378, 907, 1512, 1551 (C++ function), 379, 908, 1513, 1552  
 sh::scripting::api::ApiSetting::\_isAdvancedSetting sh::scripting::api::ApiSubmenuItem::initializ (C++ member), 378, 907, 1512, 1551 (C++ function), 379, 908, 1513, 1552  
 sh::scripting::api::ApiSetting::\_isGlobal sh::scripting::api::ApiSubmenuItem::initializ (C++ member), 378, 907, 1512, 1552 (C++ function), 380, 908, 1514, 1553  
 sh::scripting::api::ApiSetting::\_isPerFileView sh::scripting::api::ApiSubmenuItem::initializ (C++ member), 378, 907, 1512, 1552 (C++ function), 380, 909, 1514, 1553  
 sh::scripting::api::ApiSetting::\_name sh::scripting::api::ApiSubmenuItem::isChecked (C++ member), 378, 907, 1512, 1551 (C++ function), 379, 908, 1513, 1552  
 sh::scripting::api::ApiSetting::\_setValuesh::scripting::api::ApiSubmenuItem::isChecked (C++ member), 378, 907, 1512, 1551 (C++ function), 379, 908, 1513, 1552  
 sh::scripting::api::ApiSetting::\_setValuesh::scripting::api::ApiSubmenuItem::isInitializ (C++ member), 378, 907, 1512, 1551 (C++ function), 380, 909, 1514, 1553  
 sh::scripting::api::ApiSetting::\_valueDescription sh::scripting::api::ApiSubmenuItem::isInitializ (C++ member), 378, 907, 1512, 1551 (C++ function), 380, 909, 1514, 1553  
 sh::scripting::api::ApiSetting::ApiSetting sh::scripting::api::ApiSubmenuItem::parentActio (C++ member), 378, 907, 1512, 1551 (C++ function), 379, 908, 1514, 1553  
 sh::scripting::api::ApiSetting::description sh::scripting::api::ApiSubmenuItem::setChecked (C++ function), 377, 906, 1512, 1551 (C++ function), 379, 908, 1513, 1552  
 sh::scripting::api::ApiSetting::description sh::scripting::api::ApiSubmenuItem::setEnabled (C++ function), 377, 906, 1512, 1551 (C++ function), 379, 908, 1513, 1552  
 sh::scripting::api::ApiSetting::getGroupDescription sh::scripting::api::ApiSubmenuItem::setIcon (C++ function), 378, 907, 1513, 1552 (C++ function), 379, 908, 1513, 1552  
 sh::scripting::api::ApiSetting::getValuesh::scripting::api::ApiSubmenuItem::setIcon (C++ function), 378, 907, 1512, 1551 (C++ function), 379, 908, 1513, 1552  
 sh::scripting::api::ApiSetting::group sh::scripting::api::ApiSubmenuItem::setSubite (C++ function), 377, 906, 1512, 1551 (C++ function), 379, 908, 1513, 1552  
 sh::scripting::api::ApiSetting::isAdvancedSetting sh::scripting::api::ApiSubmenuItem::setText (C++ function), 378, 906, 1512, 1551 (C++ function), 379, 908, 1513, 1552  
 sh::scripting::api::ApiSetting::isGlobal sh::scripting::api::ApiSubmenuItem::setVisible (C++ function), 378, 907, 1512, 1551 (C++ function), 379, 908, 1513, 1553  
 sh::scripting::api::ApiSetting::isPerFileView sh::scripting::api::ApiSubmenuItem::setVisible (C++ function), 378, 907, 1512, 1551 (C++ function), 379, 908, 1513, 1553  
 sh::scripting::api::ApiSetting::isPerFileView sh::scripting::api::ApiSubmenuItem::subitems (C++ function), 378, 907, 1512, 1551 (C++ function), 379, 908, 1513, 1552  
 sh::scripting::api::ApiSetting::isPerFileView sh::scripting::api::ApiSubmenuItem::subitems (C++ function), 378, 907, 1512, 1551 (C++ function), 380, 909, 1514, 1553  
 sh::scripting::api::ApiSetting::name sh::scripting::api::ApiSubmenuItem::subitems (C++ function), 377, 906, 1512, 1551 (C++ function), 380, 909, 1514, 1553  
 sh::scripting::api::ApiSetting::setValuesh::scripting::api::ApiSubmenuItem::text (C++ function), 378, 907, 1512, 1551 (C++ function), 379, 908, 1513, 1552  
 sh::scripting::api::ApiSetting::setValuesh::scripting::api::ApiSubmenuItem::text (C++ function), 378, 907, 1512, 1551 (C++ function), 379, 908, 1513, 1552  
 sh::scripting::api::ApiSetting::valueDescription sh::scripting::api::ApiSubmenuItem::visible (C++ function), 378, 907, 1512, 1551 (C++ function), 379, 908, 1513, 1553  
 sh::scripting::api::ApiSetting::valueDescription sh::scripting::api::ApiThread (C++ class), (C++ function), 378, 907, 1512, 1551 380, 909, 1514, 1553  
 sh::scripting::api::ApiSubmenuItem sh::scripting::api::ApiThread::\_run (C++ class), 379, 907, 1513, 1552 (C++ member), 380, 909, 1514, 1554  
 sh::scripting::api::ApiSubmenuItem sh::scripting::api::ApiThread::ApiThread (C++ member), 380, 909, 1514, 1553 (C++ function), 380, 909, 1514, 1553  
 sh::scripting::api::ApiSubmenuItem sh::scripting::api::ApiThread::ApiThread (C++ member), 380, 909, 1514, 1553 (C++ function), 380, 909, 1514, 1553  
 sh::scripting::api::ApiSubmenuItem sh::scripting::api::ApiThread::start (C++ function), 379, 908, 1514, 1553 (C++ function), 380, 909, 1514, 1553  
 sh::scripting::api::ApiSubmenuItem sh::scripting::api::ApiThread::start (C++ function), 379, 908, 1514, 1553 (C++ function), 380, 909, 1514, 1553  
 sh::scripting::api::ApiSubmenuItem sh::scripting::api::ApiThumbnailProvider (C++ function), 379, 908, 1513, 1552 (C++ class), 380, 909, 1514, 1554  
 sh::scripting::api::ApiSubmenuItem sh::scripting::api::ApiThumbnailProvider (C++ function), 379, 908, 1513, 1552 (C++ class), 380, 909, 1514, 1554  
 sh::scripting::api::ApiSubmenuItem sh::scripting::api::ApiThumbnailProvider::\_getThum (C++ function), 380, 909, 1514, 1553 (C++ member), 381, 910, 1515, 1554  
 sh::scripting::api::ApiSubmenuItem sh::scripting::api::ApiThumbnailProvider::\_getThum (C++ function), 380, 909, 1514, 1553 (C++ member), 381, 910, 1515, 1554  
 sh::scripting::api::ApiSubmenuItem sh::scripting::api::ApiThumbnailProvider::ApiThumbn (C++ function), 379, 908, 1513, 1552 (C++ function), 381, 909, 1515, 1554



sh::scripting::api::ApiThumbnailProviders::getThumbnail (C++ function), 381, 909, 1515, 1554  
 sh::scripting::api::ApiTimer (C++ class), 381, 910, 1515, 1554  
 sh::scripting::api::ApiTimer::\_isexecuting (C++ member), 381, 910, 1515, 1554  
 sh::scripting::api::ApiTimer::\_mutex (C++ member), 381, 910, 1515, 1554  
 sh::scripting::api::ApiTimer::\_run (C++ member), 381, 910, 1515, 1554  
 sh::scripting::api::ApiTimer::\_runningTime (C++ member), 381, 910, 1515, 1554  
 sh::scripting::api::ApiTimer::\_timer (C++ member), 381, 910, 1515, 1554  
 sh::scripting::api::ApiTimer::ApiTimer (C++ function), 381, 910, 1515, 1554  
 sh::scripting::api::ApiTimer::start (C++ function), 381, 910, 1515, 1554  
 sh::scripting::api::ApiTimer::stop (C++ function), 381, 910, 1515, 1554  
 sh::scripting::Api::classes (C++ function), 343, 874, 1479  
 sh::scripting::Api::getClass (C++ function), 343, 874, 1479  
 sh::scripting::Api::getClassByIdName (C++ function), 343, 874, 1479  
 sh::scripting::Api::registerClass (C++ function), 343, 874, 1480  
 sh::scripting::Api::registerMethod (C++ function), 343, 874, 1480  
 sh::scripting::Api::registerRootObjectMembers (C++ function), 343, 874, 1480  
 sh::scripting::Api::rootMember (C++ function), 343, 874, 1479  
 sh::scripting::Api::rootMemberClass (C++ function), 343, 874, 1479  
 sh::scripting::Api::rootMembers (C++ function), 343, 874, 1479  
 sh::scripting::ApiClass (C++ class), 344, 874, 1480  
 sh::scripting::ApiClass::\_methods (C++ member), 344, 875, 1480  
 sh::scripting::ApiClass::\_name (C++ member), 344, 875, 1480  
 sh::scripting::ApiClass::~ApiClass (C++ function), 344, 875, 1480  
 sh::scripting::ApiClass::addMethod (C++ function), 344, 875, 1480  
 sh::scripting::ApiClass::ApiClass (C++ function), 344, 875, 1480  
 sh::scripting::ApiClass::getMethod (C++ function), 344, 875, 1480  
 sh::scripting::ApiClass::methods (C++ function), 344, 875, 1480  
 sh::scripting::ApiClass::name (C++ function), 344, 875, 1480  
 sh::scripting::ApiMethod (C++ class), 344, 875, 1480  
 sh::scripting::ApiMethod::ApiMethod (C++ function), 344, 875, 1480  
 sh::scripting::ApiMethod::name (C++ function), 344, 875, 1480  
 sh::scripting::pythoninterop (C++ type), 910, 1515, 1554  
 sh::scripting::pythoninterop::\_PyObjectFunctionToNative (C++ class), 389, 911, 1516, 1556  
 sh::scripting::pythoninterop::\_PyObjectFunctionToNativeArg, Args...> (C++ class), 389, 912, 1517, 1556  
 sh::scripting::pythoninterop::\_PyObjectFunctionToNativeArg, Args...>::setTupleValue (C++ function), 389, 912, 1517, 1556  
 sh::scripting::pythoninterop::\_PyObjectFunctionToNative (C++ class), 389, 911, 1516, 1556  
 sh::scripting::pythoninterop::\_PyObjectFunctionToNative (C++ function), 389, 912, 1517, 1556  
 sh::scripting::pythoninterop::\_PyObjectFunctionToNative (C++ class), 389, 912, 1517, 1556  
 sh::scripting::pythoninterop::\_PyObjectFunctionToNative (C++ function), 390, 912, 1517, 1556  
 sh::scripting::pythoninterop::\_PyObjectFunctionToNative (C++ function), 389, 912, 1517, 1556  
 sh::scripting::pythoninterop::\_PyObjectFunctionToNative (C++ member), 390, 912, 1517, 1556  
 sh::scripting::pythoninterop::\_PyObjectFunctionToNative (C++ function), 389, 912, 1517, 1556  
 sh::scripting::pythoninterop::CallNativeFunctionWith (C++ function), 911, 1516, 1555  
 sh::scripting::pythoninterop::CallNativeHelperFunction (C++ function), 911, 1516, 1556  
 sh::scripting::pythoninterop::CallNativeHelperMethod (C++ struct), 912, 1517, 1556, 1881  
 sh::scripting::pythoninterop::CallNativeMethodWith (C++ struct), 913, 1518, 1557, 1882  
 sh::scripting::pythoninterop::Converters (C++ class), 381, 913, 1518, 1557  
 sh::scripting::pythoninterop::Converters::getBoolFrom (C++ function), 382, 913, 1518, 1558  
 sh::scripting::pythoninterop::Converters::getByteArray (C++ function), 382, 913, 1518, 1558  
 sh::scripting::pythoninterop::Converters::getDouble (C++ function), 381, 913, 1518, 1558  
 sh::scripting::pythoninterop::Converters::getInt64 (C++ function), 381, 913, 1518, 1558  
 sh::scripting::pythoninterop::Converters::getIntFrom (C++ function), 381, 913, 1518, 1558  
 sh::scripting::pythoninterop::Converters::getPointer (C++ function), 382, 913, 1518, 1558

```

sh::scripting::pythoninterop::Converters$shgessPyObjectFromPyObjectFromPyBorrowedArray::getSharedPointerFromM
(C++ function), 382, 914, 1519, 1558 (C++ function), 910, 1515, 1555
sh::scripting::pythoninterop::Converters$shgessPyObjectFromPyObjectFromPyBorrowedArray::getSharedPointerFromM
(C++ function), 382, 914, 1519, 1558 (C++ function), 910, 1515, 1555
sh::scripting::pythoninterop::Converters$shgessPyObjectFromPyObjectFromPyBorrowedArray::invokePython
(C++ function), 382, 914, 1519, 1558 (C++ function), 911, 1516, 1555
sh::scripting::pythoninterop::Converters$shgessPyObjectFromPyObjectFromPyBorrowedArray::isPyNone
(C++ function), 382, 914, 1519, 1558 (C++ function), 910, 1515, 1555
sh::scripting::pythoninterop::Converters$shgessPyObjectFromPyObjectFromPyBorrowedArray::NativeToPyObject
(C++ function), 382, 914, 1519, 1558 (C++ class), 382, 915, 1520, 1559
sh::scripting::pythoninterop::Converters$shgessPyObjectFromPyObjectFromPyBorrowedArray::NativeToPyObject<bool>
(C++ function), 382, 914, 1519, 1558 (C++ class), 383, 915, 1520, 1559
sh::scripting::pythoninterop::Converters$shgessPyObjectFromPyObjectFromPyBorrowedArray::NativeToPyObject<bool>
(C++ function), 382, 914, 1519, 1558 (C++ function), 384, 915, 1520, 1559
sh::scripting::pythoninterop::Converters$shgessPyObjectFromPyObjectFromPyBorrowedArray::NativeToPyObject<double>
(C++ function), 382, 914, 1519, 1558 (C++ class), 384, 915, 1520, 1559
sh::scripting::pythoninterop::Converters$shgessPyObjectFromPyObjectFromPyBorrowedArray::NativeToPyObject<double>
(C++ function), 382, 914, 1519, 1558 (C++ function), 384, 915, 1520, 1559
sh::scripting::pythoninterop::Converters$shgessPyObjectFromPyObjectFromPyBorrowedArray::NativeToPyObject<int>
(C++ function), 381, 913, 1518, 1558 (C++ class), 384, 915, 1520, 1559
sh::scripting::pythoninterop::Converters$shgessPyObjectFromPyObjectFromPyBorrowedArray::NativeToPyObject<int>
(C++ function), 381, 913, 1518, 1558 (C++ function), 384, 915, 1520, 1560
sh::scripting::pythoninterop::Converters$shgessPyObjectFromPyObjectFromPyBorrowedArray::NativeToPyObject<QByteArray>
(C++ function), 381, 913, 1518, 1558 (C++ class), 382, 915, 1520, 1560
sh::scripting::pythoninterop::FunctionBindTuple$shgessPyObjectFromPyObjectFromPyBorrowedArray::NativeToPyObject<QByteArray>
(C++ struct), 914, 1519, 1558, 1883 (C++ function), 382, 915, 1520, 1560
sh::scripting::pythoninterop::FunctionBindTuple$shgessPyObjectFromPyObjectFromPyBorrowedArray::NativeToPyObject<QByteArray>
0> (C++ struct), 914, 1519, 1558, 1883 (C++ class), 384, 915, 1520, 1560
sh::scripting::pythoninterop::FunctionBindTuple$shgessPyObjectFromPyObjectFromPyBorrowedArray::NativeToPyObject<QByteArray>
0>::applyTuple (C++ function), 914, (C++ function), 384, 916, 1521, 1560
1519, 1558, 1883 sh::scripting::pythoninterop::NativeToPyObject<QLinkedList>
sh::scripting::pythoninterop::FunctionBindTuple$shgessPyObjectFromPyObjectFromPyBorrowedArray::NativeToPyObject<QLinkedList>
N> (C++ struct), 914, 1519, 1558, 1883 sh::scripting::pythoninterop::NativeToPyObject<QLinkedList>
sh::scripting::pythoninterop::FunctionBindTuple$shgessPyObjectFromPyObjectFromPyBorrowedArray::NativeToPyObject<QLinkedList>
N>::applyTuple (C++ function), 914, sh::scripting::pythoninterop::NativeToPyObject<QMap>
1519, 1559, 1883 V>> (C++ class), 383, 916, 1521, 1560
sh::scripting::pythoninterop::FunctionBindTuple$shgessPyObjectFromPyObjectFromPyBorrowedArray::NativeToPyObject<QMap>
0> (C++ struct), 914, 1519, 1559, 1884 V>>::toPyObject (C++ function), 383,
915, 1521, 1560
sh::scripting::pythoninterop::FunctionBindTuple$shgessPyObjectFromPyObjectFromPyBorrowedArray::NativeToPyObject<QMap>
0>::applyTuple (C++ function), 914, sh::scripting::pythoninterop::NativeToPyObject<QMap>
1519, 1559, 1884 (C++ class), 383, 916, 1521, 1560
sh::scripting::pythoninterop::FunctionBindTuple$shgessPyObjectFromPyObjectFromPyBorrowedArray::NativeToPyObject<QMap>
N> (C++ struct), 914, 1519, 1559, 1884 (C++ function), 383, 916, 1521, 1560
sh::scripting::pythoninterop::FunctionBindTuple$shgessPyObjectFromPyObjectFromPyBorrowedArray::NativeToPyObject<QMap>
N>::applyTuple (C++ function), 915, (C++ class), 384, 916, 1521, 1560
1520, 1559, 1884 sh::scripting::pythoninterop::NativeToPyObject<QMap>
sh::scripting::pythoninterop::getApiClass (C++ function), 384, 916, 1521, 1561
(C++ function), 910, 1516, 1555 sh::scripting::pythoninterop::NativeToPyObject<QMap>
sh::scripting::pythoninterop::getPointerFromSharedPyObjectFromPyBorrowedArray::getPointerFromSharedPyObjectFromPyBorrowedArray
(C++ function), 910, 1515, 1555 (C++ function), 383, 916, 1521, 1561
sh::scripting::pythoninterop::getPyNone (C++ function), 910, 1515, 1555 (C++ function), 383, 916, 1521, 1561
sh::scripting::pythoninterop::getShallotPyObjectFromPyObjectFromPyBorrowedArray::getShallotPyObjectFromPyObjectFromPyBorrowedArray
(C++ function), 910, 1515, 1555 (C++ class), 385
sh::scripting::pythoninterop::NativeToPyObject<std::string>
sh::scripting::pythoninterop::NativeToPyObject<std::string>

```

Index 1983

sh::scripting::pythoninterop::PyObjectToNative<C++ member>, 346, 876, 1482  
     (C++ class), 387, 919, 1524, 1564      sh::scripting::PythonScriptInterpreter::\_natives  
 sh::scripting::pythoninterop::PyObjectToNative<C++ member>, 345, 876, 1481  
     (C++ function), 387, 919, 1524, 1564      sh::scripting::PythonScriptInterpreter::\_ThreadAbort  
 sh::scripting::pythoninterop::PyObjectToNative<C++ member>, 345, 876, 1481  
     (C++ class), 387, 919, 1524, 1564      sh::scripting::PythonScriptInterpreter::~~PythonScriptInterpreter  
 sh::scripting::pythoninterop::PyObjectToNative<C++ function>, 345, 875, 1481  
     (C++ function), 387, 920, 1525, 1564      sh::scripting::PythonScriptInterpreter::api  
 sh::scripting::pythoninterop::PyObjectTupleToNative<C++ member>, 346, 876, 1482  
     (C++ struct), 920, 1525, 1564, 1884      sh::scripting::PythonScriptInterpreter::apiModuleDe  
 sh::scripting::pythoninterop::PyObjectTupleToNative<C++ member>, 346, 876, 1482  
     T, Args...> (C++ struct), 920, 1525, 1564, 1884      sh::scripting::PythonScriptInterpreter::doInitializ  
     (C++ function), 345, 875, 1481  
 sh::scripting::pythoninterop::PyObjectTupleToNativeToPythonScriptInterpreter::execute  
     T, Args...>::toNativeTuple (C++      (C++ function), 345, 875, 1481  
     function), 920, 1525, 1564, 1884      sh::scripting::PythonScriptInterpreter::filesuffix  
 sh::scripting::pythoninterop::PyObjectTupleToNative<C++ function>, 345, 875, 1481  
     (C++ struct), 920, 1525, 1564, 1884      sh::scripting::PythonScriptInterpreter::getApiModul  
 sh::scripting::pythoninterop::PyObjectTupleToNative<C++ function>, 346, 876, 1481  
     (C++ function), 920, 1525, 1564, 1885      sh::scripting::PythonScriptInterpreter::initialize  
 sh::scripting::pythoninterop::PythonToC\_ExecutedC++ function), 345, 875, 1481  
     (C++ function), 911, 1516, 1555      sh::scripting::PythonScriptInterpreter::isAlive  
 sh::scripting::pythoninterop::PyTupleCount (C++ function), 345, 875, 1481  
     (C++ function), 910, 1516, 1555      sh::scripting::PythonScriptInterpreter::isAvailable  
 sh::scripting::pythoninterop::PyTupleGetItem (C++ function), 345, 875, 1481  
     (C++ function), 910, 1516, 1555      sh::scripting::PythonScriptInterpreter::modtracebac  
 sh::scripting::pythoninterop::PyTupleNew (C++ member), 346, 876, 1482  
     (C++ function), 911, 1516, 1555      sh::scripting::PythonScriptInterpreter::PythonScript  
 sh::scripting::pythoninterop::PyTupleSetItem (C++ function), 346, 876, 1481  
     (C++ function), 911, 1516, 1555      sh::scripting::PythonScriptInterpreter::registerMet  
 sh::scripting::pythoninterop::ScriptedMethodProc(C++ function), 345, 876, 1481  
     (C++ struct), 920, 1525, 1564, 1885      sh::scripting::PythonScriptInterpreter::shutdown  
 sh::scripting::pythoninterop::ScriptedMethodProc(C++ function), 345, 875, 1481  
     std::function<R(Args...)> T::\*,      sh::scripting::PythonScriptInterpreter::throwNative  
     MF> (C++ struct), 1885      (C++ function), 345, 876, 1481  
 sh::scripting::pythoninterop::ScriptedMethodProc(C++ function), 345, 875, 1481  
     std::function<R(Args...)> T::\*,      sh::scripting::ScriptingEngine (C++  
     MF>::pyObjectFunctionSetImplementation      class), 346, 876, 1482  
     (C++ function), 1885      sh::scripting::ScriptingEngine::\_interpreters  
     (C++ member), 347, 877, 1482  
 sh::scripting::pythoninterop::ScriptedMethodProc(C++ function), 345, 875, 1481  
     std::function<R(Args...)> T::\*,      sh::scripting::ScriptingEngine::ActionPluginLoadCra  
     MF>::setImplementation (C++ func-      (C++ class), 347, 349, 877, 1482  
     tion), 1885      (C++ function), 348, 350, 878, 1483  
 sh::scripting::pythoninterop::StringMap (C++ type), 910, 1515, 1555      sh::scripting::ScriptingEngine::ActionPluginLoadCra  
     (C++ function), 347, 350, 877, 1482  
 sh::scripting::PythonScriptInterpreter (C++ class), 345, 875, 1480      sh::scripting::ScriptingEngine::ActionPluginLoadCra  
     (C++ function), 348, 351, 878, 1484  
 sh::scripting::PythonScriptInterpreter::shutdown      sh::scripting::ScriptingEngine::ActionPluginLoadCra  
     (C++ member), 345, 876, 1481      (C++ function), 347, 350, 877, 1483  
 sh::scripting::PythonScriptInterpreter::handleException      sh::scripting::ScriptingEngine::ActionPluginLoadCra  
     (C++ member), 345, 876, 1481      (C++ function), 347, 350, 877, 1483  
 sh::scripting::PythonScriptInterpreter::ExceptionHandler      sh::scripting::ScriptingEngine::ActionPluginLoadCra  
     (C++ member), 345, 876, 1481      (C++ function), 347, 350, 877, 1482  
 sh::scripting::PythonScriptInterpreter::methods      sh::scripting::ScriptingEngine::ActionPluginLoadCra









sh::search::criteria::FileContentSearchCriteria::FileContentSearchCriteria(MtimeSearchCriteria::SubstringDistance (C++ function), 405, 936, 1580, 1588 (C++ class), 407, 938, 1581, 1589

sh::search::criteria::FileContentSearchCriteria::FileContentSearchCriteria(MtimeSearchCriteria::create (C++ function), 404, 935, 1579, 1587 (C++ function), 407, 938, 1582, 1590

sh::search::criteria::FileContentSearchCriteria::FileContentSearchCriteria(MtimeSearchCriteria::deserialize (C++ member), 405, 936, 1580, 1588 (C++ function), 407, 938, 1582, 1590

sh::search::criteria::FileContentSearchCriteria::FileContentSearchCriteria(MtimeSearchCriteria::doInit (C++ function), 404, 935, 1579, 1587 (C++ function), 407, 938, 1582, 1590

sh::search::criteria::FilenameSearchCriteria::FilenameSearchCriteria(MtimeSearchCriteria::doShutdown (C++ class), 405, 936, 1580, 1588 (C++ function), 407, 938, 1582, 1590

sh::search::criteria::FilenameSearchCriteria::FilenameSearchCriteria(MtimeSearchCriteria::editor (C++ function), 406, 937, 1581, 1589 (C++ function), 407, 938, 1582, 1590

sh::search::criteria::FilenameSearchCriteria::FilenameSearchCriteria(MtimeSearchCriteria::factory (C++ function), 406, 937, 1581, 1589 (C++ function), 407, 938, 1582, 1590

sh::search::criteria::FilenameSearchCriteria::FilenameSearchCriteria(MtimeSearchCriteria::from (C++ function), 406, 937, 1581, 1589 (C++ member), 407, 938, 1582, 1590

sh::search::criteria::FilenameSearchCriteria::FilenameSearchCriteria(MtimeSearchCriteria::match (C++ function), 406, 937, 1581, 1589 (C++ function), 407, 938, 1582, 1590

sh::search::criteria::FilenameSearchCriteria::FilenameSearchCriteria(MtimeSearchCriteria::MtimeSearchCriteria (C++ function), 406, 937, 1581, 1589 (C++ function), 407, 938, 1582, 1590

sh::search::criteria::FilenameSearchCriteria::FilenameSearchCriteria(MtimeSearchCriteria::serial (C++ function), 406, 937, 1581, 1589 (C++ function), 407, 938, 1582, 1590

sh::search::criteria::FilenameSearchCriteria::FilenameSearchCriteria(MtimeSearchCriteria::to (C++ function), 406, 937, 1581, 1589 (C++ member), 407, 938, 1582, 1590

sh::search::criteria::FilenameSearchCriteria::FilenameSearchCriteria(MtimeSearchCriteria::valued (C++ function), 406, 937, 1581, 1589 (C++ function), 407, 938, 1582, 1590

sh::search::criteria::FilenameSearchCriteria::FilenameSearchCriteria(C++ class), 390, 920, 1565 (C++ function), 406, 937, 1581, 1589 sh::search::Search::\_config (C++ member), 391, 921, 1566

sh::search::criteria::FilenameSearchCriteria::FilenameSearchCriteria(C++ function), 406, 937, 1581, 1589 sh::search::Search::SubstringDistance (C++ function), 406, 937, 1581, 1589 sh::search::Search::\_dirs (C++ member), 391, 921, 1566

sh::search::criteria::FilenameSearchCriteria::FilenameSearchCriteria(C++ function), 406, 937, 1581, 1589 sh::search::Search::\_eurl (C++ member), 391, 921, 1566

sh::search::criteria::FilenameSearchCriteria::FilenameSearchCriteria(C++ enum), 405, 936, 1580, 1588 sh::search::Search::\_isSearchAgain (C++ member), 391, 921, 1566

sh::search::criteria::FilenameSearchCriteria::FilenameSearchCriteria(C++ member), 406, 937, 1581, 1589 sh::search::Search::\_isSearching (C++ member), 391, 921, 1566

sh::search::criteria::FilenameSearchCriteria::FilenameSearchCriteria(C++ enumerator), 405, 936, 1580, 1588 sh::search::Search::\_isSearchingMutex (C++ member), 391, 921, 1566

sh::search::criteria::FilenameSearchCriteria::FilenameSearchCriteria(C++ enumerator), 405, 936, 1580, 1588 sh::search::Search::\_searchnode (C++ member), 391, 921, 1566

sh::search::criteria::FilenameSearchCriteria::FilenameSearchCriteria(C++ enumerator), 405, 936, 1580, 1588 sh::search::Search::~Search (C++ function), 390, 921, 1565

sh::search::criteria::FilenameSearchCriteria::FilenameSearchCriteria(C++ enumerator), 405, 936, 1580, 1588 sh::search::Search::eurl (C++ function), 390, 921, 1565

sh::search::criteria::FilenameSearchCriteria::FilenameSearchCriteria(C++ member), 406, 938, 1581, 1589 sh::search::Search::getDirItem (C++ function), 390, 921, 1565

sh::search::criteria::FilenameSearchCriteria::FilenameSearchCriteria(C++ function), 406, 937, 1581, 1589 sh::search::Search::HeinPartsSubstringDistance (C++ function), 390, 921, 1565

sh::search::criteria::FilenameSearchCriteria::FilenameSearchCriteria(C++ function), 406, 937, 1581, 1589 sh::search::Search::insertFileItem (C++ function), 390, 921, 1565

sh::search::criteria::FilenameSearchCriteria::FilenameSearchCriteria(C++ function), 406, 937, 1581, 1589 sh::search::Search::isSearching (C++ member), 406, 937, 1581, 1589

sh::search::criteria::FilenameSearchCriteria::FilenameSearchCriteria(C++ member), 406, 937, 1581, 1589 sh::search::Search::isSearchingChanged (C++ function), 390, 921, 1565

sh::search::criteria::FilenameSearchCriteria::FilenameSearchCriteria(C++ function), 406, 937, 1581, 1589 sh::search::Search::Search (C++ function), 390, 921, 1565



390, 921, 1565

sh::search::Search::search (C++ function), 390, 921, 1565

sh::search::Search::searchnode (C++ function), 390, 921, 1565

sh::search::SearchConfiguration (C++ class), 391, 921, 1566

sh::search::SearchConfiguration::criteria (C++ member), 391, 922, 1566

sh::search::SearchConfiguration::deserialize (C++ function), 391, 922, 1566

sh::search::SearchConfiguration::SearchConfiguration (C++ function), 391, 922, 1566

sh::search::SearchConfiguration::serialize (C++ function), 391, 922, 1566

sh::search::SearchConfiguration::showAsText (C++ member), 391, 922, 1566

sh::search::SearchCriterion (C++ class), 391, 922, 1566

sh::search::SearchCriterion::~~SearchCriterion (C++ function), 392, 922, 1567

sh::search::SearchCriterion::deserialize (C++ function), 392, 923, 1567

sh::search::SearchCriterion::factory (C++ function), 392, 922, 1567

sh::search::SearchCriterion::match (C++ function), 392, 922, 1567

sh::search::SearchCriterion::SearchCriterion (C++ function), 392, 922, 1567

sh::search::SearchCriterion::serialize (C++ function), 392, 922, 1567

sh::search::SearchCriterion::valueDescription (C++ function), 392, 922, 1567

sh::search::SearchCriterionFactory (C++ class), 392, 923, 1567

sh::search::SearchCriterionFactory::~~SearchCriterionFactory (C++ function), 393, 923, 1568

sh::search::SearchCriterionFactory::construct (C++ function), 392, 923, 1567

sh::search::SearchCriterionFactory::description (C++ function), 392, 923, 1567

sh::search::SearchCriterionFactory::editor (C++ function), 392, 923, 1567

sh::search::SearchCriterionFactory::editorUpdate (C++ function), 392, 923, 1567

sh::search::SearchCriterionFactory::isVisibleFor (C++ function), 392, 923, 1567

sh::search::SearchCriterionFactory::key (C++ function), 392, 923, 1567

sh::search::SearchCriterionFactory::SearchCriterionFactory (C++ function), 392, 923, 1567

sh::search::SearchCriterionFactoryFromFunction (C++ class), 393, 923, 1568

sh::search::SearchCriterionFactoryFromFunction::~SearchCriterionFactoryFromFunction (C++ member), 394, 924, 1568

sh::search::SearchCriterionFactoryFromFunction::SearchCriterionFactoryFromFunction (C++ member), 394, 924, 1568

sh::search::SearchCriterionFactoryFromFunction::SearchCriterionFactoryFromFunction (C++ member), 394, 924, 1568

sh::search::SearchCriterionFactoryFromFunction::key (C++ member), 394, 924, 1568

sh::search::SearchCriterionFactoryFromFunction::criteria (C++ function), 393, 924, 1568

sh::search::SearchCriterionFactoryFromFunction::deserialize (C++ function), 393, 924, 1568

sh::search::SearchCriterionFactoryFromFunction::editor (C++ function), 393, 924, 1568

sh::search::SearchCriterionFactoryFromFunction::editorUpdate (C++ function), 393, 924, 1568

sh::search::SearchCriterionFactoryFromFunction::isVisibleFor (C++ function), 393, 924, 1568

sh::search::SearchCriterionFactoryFromFunction::key (C++ function), 393, 924, 1568

sh::search::SearchCriterionFactoryFromFunction::SearchCriterionFactoryFromFunction (C++ function), 393, 924, 1568

sh::search::SearchFilesystemHandler (C++ class), 394, 924, 1569

sh::search::SearchFilesystemHandler::canCreateDirectory (C++ function), 394, 396, 925, 927, 1569, 1571

sh::search::SearchFilesystemHandler::canCreateFile (C++ function), 394, 396, 925, 927, 1569, 1571

sh::search::SearchFilesystemHandler::canCreateLink (C++ function), 394, 396, 925, 927, 1569, 1571

sh::search::SearchFilesystemHandler::canDeleteItem (C++ function), 395, 396, 925, 927, 1569, 1571

sh::search::SearchFilesystemHandler::canGetFileContent (C++ function), 394, 396, 925, 927, 1569, 1571

sh::search::SearchFilesystemHandler::canRenameItem (C++ function), 395, 397, 925, 927, 1569, 1571

sh::search::SearchFilesystemHandler::configureItems (C++ function), 395, 926, 1570

sh::search::SearchFilesystemHandler::createDirectory (C++ function), 394, 396, 925, 927, 1569, 1571

sh::search::SearchFilesystemHandler::createFile (C++ function), 394, 396, 925, 927, 1569, 1571

sh::search::SearchFilesystemHandler::createLink (C++ function), 394, 396, 925, 927, 1569, 1571

sh::search::SearchFilesystemHandler::customizeUi (C++ function), 395, 397, 925, 928, 1570, 1571

1572 (C++ function), 397, 928, 1572  
sh::search::SearchFilesystemHandler::deleteSearchCriteriaEmptyFilesystemHandler::SearchFilesystemHandler (C++ function), 396, 927, 1571 (C++ function), 398, 929, 1573  
sh::search::SearchFilesystemHandler::deleteItemsh::search::SearchFilesystemHandler::setCustomAttributes (C++ function), 395, 396, 925, 927, 1569, (C++ function), 396, 927, 1571  
1571 sh::search::SearchFilesystemHandler::setExtendedAttributes (C++ function), 395, 926, 1570  
sh::search::SearchFilesystemHandler::doShutdown (C++ function), 397, 928, 1572 sh::search::SearchFilesystemHandler::setMtime (C++ function), 394, 395, 925, 926, 1569, (C++ function), 395, 397, 925, 928, 1569, 1570  
1572 sh::search::SearchFilesystemHandler::shutdown (C++ function), 397, 928, 1572  
sh::search::SearchFilesystemHandler::getCustomAttributes (C++ function), 396, 926, 1570 sh::search::SearchFilesystemHandler::viewEnteredDirectories (C++ function), 397, 928, 1572  
sh::search::SearchFilesystemHandler::getExtendedAttributes (C++ function), 395, 926, 1570 sh::search::SearchFilesystemHandler::viewLeftDirectories (C++ function), 395, 926, 1570  
sh::search::SearchFilesystemHandler::getExtendedAttributes (C++ function), 397, 928, 1572 sh::search::SearchManager (C++ class), 398, 929, 1573  
sh::search::SearchFilesystemHandler::getFileContents (C++ function), 394, 396, 925, 927, 1569, sh::search::SearchManager::\_factories (C++ member), 399, 930, 1574  
1571 sh::search::SearchFilesystemHandler::getLinkTargetsh::search::SearchManager::\_searches (C++ member), 399, 930, 1574  
sh::search::SearchFilesystemHandler::getLinkTargetsh::search::SearchManager::\_weak\_searches (C++ member), 399, 930, 1574  
sh::search::SearchFilesystemHandler::getMimeType (C++ function), 394, 395, 925, 926, 1569, sh::search::SearchManager::~~SearchManager (C++ function), 398, 929, 1573  
1570 sh::search::SearchFilesystemHandler::getTime (C++ function), 394, 395, 925, 926, 1569, sh::search::SearchManager::addFactory (C++ function), 398, 929, 1573  
1570 sh::search::SearchFilesystemHandler::doShutdown (C++ function), 398, 929, 1573  
sh::search::SearchFilesystemHandler::getSize (C++ function), 394, 395, 925, 926, 1569, sh::search::SearchManager::factories (C++ function), 398, 929, 1573  
1570 sh::search::SearchFilesystemHandler::getType (C++ function), 394, 395, 925, 926, 1569, sh::search::SearchManager::findSearchCriteria (C++ function), 398, 929, 1573  
1570 sh::search::SearchFilesystemHandler::isAlive (C++ function), 397, 928, 1572 sh::search::SearchManager::isAlive (C++ function), 398, 929, 1573  
sh::search::SearchFilesystemHandler::isAlive (C++ function), 397, 928, 1572 sh::search::SearchManager::REGISTER\_SEARCHCRITERION (C++ member), 398, 929, 1573  
sh::search::SearchFilesystemHandler::isWellKnown (C++ function), 397, 928, 1572 sh::search::SearchManager::REGISTER\_SEARCHCRITERION (C++ member), 398, 929, 1573  
sh::search::SearchFilesystemHandler::itemlist (C++ member), 398, 929, 1573 sh::search::SearchManager::REGISTER\_SEARCHCRITERION (C++ member), 398, 929, 1573  
sh::search::SearchFilesystemHandler::listExtendedAttributessh::search::SearchManager::removeSearchCriteria (C++ function), 395, 926, 1570 (C++ function), 398, 929, 1573  
sh::search::SearchFilesystemHandler::modelsh::search::SearchManager::requestSearchCriteria (C++ function), 397, 928, 1572 (C++ function), 398, 929, 1573  
sh::search::SearchFilesystemHandler::removeExtendedAttributessh::search::SearchManager::SearchManager (C++ function), 396, 926, 1570 (C++ function), 399, 930, 1574  
sh::search::SearchFilesystemHandler::renameItemsh::search::SearchManager::shutdown (C++ function), 395, 397, 925, 927, 1569, (C++ function), 398, 929, 1573  
1572 sh::settings (C++ type), 938, 1590  
sh::search::SearchFilesystemHandler::requestResolvedLinksh::settings::common (C++ type), 944, 1596, 1604  
sh::search::SearchFilesystemHandler::searchsh::settings::common::FileDetailsPanelVisible



sh::settings::common::JumpbarMode::isGlobal (C++ function), 419, 949, 1601, 1609  
 (C++ function), 418, 948, 1600, 1608 sh::settings::common::ShowHiddenFiles::setValue  
 sh::settings::common::JumpbarMode::isPerFileview (C++ function), 419, 420, 949, 1601, 1609  
 (C++ function), 418, 948, 1600, 1608 sh::settings::common::ShowHiddenFiles::ShowHiddenFiles  
 sh::settings::common::JumpbarMode::JumpbarMode (C++ function), 419, 949, 1601, 1609  
 (C++ function), 418, 948, 1599, 1608 sh::settings::common::ShowHiddenFiles::valueDescription  
 sh::settings::common::JumpbarMode::name (C++ function), 420, 949, 1601, 1609  
 (C++ function), 418, 948, 1599, 1608 sh::settings::common::ShowTree (C++  
 sh::settings::common::JumpbarMode::setValue class), 420, 950, 1601, 1610  
 (C++ function), 418, 948, 1599, 1600, 1608 sh::settings::common::ShowTree::description  
 sh::settings::common::JumpbarMode::valueDescription (C++ function), 420, 950, 1602, 1610  
 (C++ function), 418, 948, 1600, 1608 sh::settings::common::ShowTree::getGroupDescription  
 sh::settings::common::NumberOfFilePanels (C++ function), 421, 950, 1602, 1610  
 (C++ class), 418, 948, 1600, 1608 sh::settings::common::ShowTree::getValue  
 sh::settings::common::NumberOfFilePanels::description (C++ function), 420, 950, 1602, 1610  
 (C++ function), 419, 948, 1600, 1608 sh::settings::common::ShowTree::group  
 sh::settings::common::NumberOfFilePanels::getGroupDescription (C++ function), 420, 950, 1602, 1610  
 (C++ function), 419, 949, 1601, 1609 sh::settings::common::ShowTree::isAdvancedSetting  
 sh::settings::common::NumberOfFilePanels::getValue (C++ function), 420, 950, 1602, 1610  
 (C++ function), 419, 948, 1600, 1608 sh::settings::common::ShowTree::isGlobal  
 sh::settings::common::NumberOfFilePanels::group (C++ function), 420, 950, 1602, 1610  
 (C++ function), 419, 948, 1600, 1608 sh::settings::common::ShowTree::isPerFileview  
 sh::settings::common::NumberOfFilePanels::isAdvancedSetting (C++ function), 420, 950, 1602, 1610  
 (C++ function), 419, 948, 1600, 1608 sh::settings::common::ShowTree::name  
 sh::settings::common::NumberOfFilePanels::isGlobal (C++ function), 420, 950, 1602, 1610  
 (C++ function), 419, 949, 1600, 1609 sh::settings::common::ShowTree::setValue  
 sh::settings::common::NumberOfFilePanels::isPerFileview (C++ function), 420, 950, 1602, 1610  
 (C++ function), 419, 949, 1600, 1609 sh::settings::common::ShowTree::ShowTree  
 sh::settings::common::NumberOfFilePanels::name (C++ function), 420, 950, 1602, 1610  
 (C++ function), 419, 948, 1600, 1608 sh::settings::common::ShowTree::valueDescription  
 sh::settings::common::NumberOfFilePanels::NumberOfFilePanels (C++ function), 420, 950, 1602, 1610  
 (C++ function), 419, 948, 1600, 1608 sh::settings::common::SizeFormattingMode  
 sh::settings::common::NumberOfFilePanels::setValue (C++ class), 421, 950, 1602, 1610  
 (C++ function), 419, 948, 949, 1600, 1608, 1609 sh::settings::common::SizeFormattingMode::description  
 sh::settings::common::NumberOfFilePanels::setValue (C++ function), 421, 951, 1602, 1611  
 (C++ function), 419, 949, 1600, 1609 sh::settings::common::SizeFormattingMode::getGroupDescription  
 sh::settings::common::ShowHiddenFiles sh::settings::common::SizeFormattingMode::getValue  
 (C++ class), 419, 949, 1601, 1609 (C++ function), 421, 951, 1602, 1611  
 sh::settings::common::ShowHiddenFiles::description sh::settings::common::SizeFormattingMode::group  
 (C++ function), 419, 949, 1601, 1609 (C++ function), 421, 951, 1602, 1611  
 sh::settings::common::ShowHiddenFiles::getGroupDescription sh::settings::common::SizeFormattingMode::isAdvancedSetting  
 (C++ function), 420, 950, 1601, 1610 (C++ function), 421, 951, 1602, 1611  
 sh::settings::common::ShowHiddenFiles::getValue sh::settings::common::SizeFormattingMode::isGlobal  
 (C++ function), 420, 949, 1601, 1609 (C++ function), 421, 951, 1603, 1611  
 sh::settings::common::ShowHiddenFiles::group sh::settings::common::SizeFormattingMode::isPerFileview  
 (C++ function), 419, 949, 1601, 1609 (C++ function), 421, 951, 1603, 1611  
 sh::settings::common::ShowHiddenFiles::isAdvancedSetting sh::settings::common::SizeFormattingMode::name  
 (C++ function), 419, 949, 1601, 1609 (C++ function), 421, 951, 1602, 1611  
 sh::settings::common::ShowHiddenFiles::isGlobal sh::settings::common::SizeFormattingMode::setValue  
 (C++ function), 420, 949, 1601, 1609 (C++ function), 421, 951, 1602, 1603, 1611  
 sh::settings::common::ShowHiddenFiles::isPerFileview sh::settings::common::SizeFormattingMode::SizeFormattingMode  
 (C++ function), 420, 949, 1601, 1609 (C++ function), 421, 951, 1602, 1611  
 sh::settings::common::ShowHiddenFiles::name sh::settings::common::SizeFormattingMode::valueDescription



(C++ function), 421, 951, 1603, 1611  
 sh::settings::common::StickyTreeView  
 (C++ class), 422, 951, 1603, 1611  
 sh::settings::common::StickyTreeView::description  
 (C++ function), 422, 951, 1603, 1611  
 sh::settings::common::StickyTreeView::getGroupDescription  
 (C++ function), 422, 952, 1604, 1612  
 sh::settings::common::StickyTreeView::getValue  
 (C++ function), 422, 951, 1603, 1611  
 sh::settings::common::StickyTreeView::group  
 (C++ function), 422, 951, 1603, 1611  
 sh::settings::common::StickyTreeView::isAdvancedSetting  
 (C++ function), 422, 951, 1603, 1611  
 sh::settings::common::StickyTreeView::isGlobal  
 (C++ function), 422, 952, 1603, 1612  
 sh::settings::common::StickyTreeView::isPerFile  
 (C++ function), 422, 952, 1603, 1612  
 sh::settings::common::StickyTreeView::name  
 (C++ function), 422, 951, 1603, 1611  
 sh::settings::common::StickyTreeView::setValue  
 (C++ function), 422, 951, 952, 1603, 1611, 1612  
 sh::settings::common::StickyTreeView::StickyTreeView  
 (C++ function), 422, 951, 1603, 1611  
 sh::settings::common::StickyTreeView::valueDescription  
 (C++ function), 422, 952, 1603, 1612  
 sh::settings::common::WindowTitle (C++ class), 422, 952, 1604, 1612  
 sh::settings::common::WindowTitle::description  
 (C++ function), 423, 952, 1604, 1612  
 sh::settings::common::WindowTitle::getGroupDescription  
 (C++ function), 423, 953, 1604, 1613  
 sh::settings::common::WindowTitle::getValue  
 (C++ function), 423, 952, 1604, 1612  
 sh::settings::common::WindowTitle::group  
 (C++ function), 423, 952, 1604, 1612  
 sh::settings::common::WindowTitle::isAdvancedSetting  
 (C++ function), 423, 952, 1604, 1612  
 sh::settings::common::WindowTitle::isGlobal  
 (C++ function), 423, 952, 1604, 1612  
 sh::settings::common::WindowTitle::isPerFile  
 (C++ function), 423, 952, 1604, 1612  
 sh::settings::common::WindowTitle::name  
 (C++ function), 423, 952, 1604, 1612  
 sh::settings::common::WindowTitle::setValue  
 (C++ function), 423, 952, 1604, 1612  
 sh::settings::common::WindowTitle::valueDescription  
 (C++ function), 423, 952, 1604, 1612  
 sh::settings::common::WindowTitle::WindowTitle  
 (C++ function), 423, 952, 1604, 1612  
 sh::settings::ProfileNode (C++ class), 407, 939, 1591  
 sh::settings::ProfileNode::\_eurl (C++ member), 408, 940, 1592  
 sh::settings::ProfileNode::\_inheritsFrom  
 (C++ member), 408, 940, 1592  
 sh::settings::ProfileNode::\_nodeId (C++ member), 408, 940, 1592  
 sh::settings::ProfileNode::\_profileName  
 (C++ member), 408, 940, 1592  
 sh::settings::ProfileNode::\_values (C++ member), 408, 940, 1592  
 sh::settings::ProfileNode::\_valuesOnlyInFileview  
 (C++ member), 408, 940, 1592  
 sh::settings::ProfileNode::\_withSubfolders  
 (C++ member), 408, 940, 1592  
 sh::settings::ProfileNode::eurl (C++ member), 408, 939, 1591  
 sh::settings::ProfileNode::getSettingNames  
 (C++ function), 408, 939, 1591  
 sh::settings::ProfileNode::getSettingOnlyInFileview  
 (C++ function), 408, 940, 1591  
 sh::settings::ProfileNode::getSettingValue  
 (C++ function), 408, 940, 1591  
 sh::settings::ProfileNode::inheritsFrom  
 (C++ function), 408, 939, 1591  
 sh::settings::ProfileNode::nodeId (C++ member), 408, 939, 1591  
 sh::settings::ProfileNode::profileName  
 (C++ function), 408, 939, 1591  
 sh::settings::ProfileNode::ProfileNode  
 (C++ function), 408, 939, 1591  
 sh::settings::ProfileNode::setEurl (C++ function), 408, 939, 1591  
 sh::settings::ProfileNode::setInheritsFrom  
 (C++ function), 408, 939, 1591  
 sh::settings::ProfileNode::setNodeId  
 (C++ function), 408, 939, 1591  
 sh::settings::ProfileNode::setProfileName  
 (C++ function), 408, 939, 1591  
 sh::settings::ProfileNode::setSetting  
 (C++ function), 408, 939, 1591  
 sh::settings::ProfileNode::setWithSubfolders  
 (C++ function), 408, 939, 1591  
 sh::settings::ProfileNode::withSubfolders  
 (C++ function), 408, 939, 1591  
 sh::settings::Setting (C++ class), 409, 940, 1592  
 sh::settings::Setting::~~Setting (C++ function), 409, 940, 1592  
 sh::settings::Setting::description (C++ function), 409, 940, 1592  
 sh::settings::Setting::getGroupDescription  
 (C++ function), 410, 941, 1593  
 sh::settings::Setting::getValue (C++ function), 409, 941, 1592  
 sh::settings::Setting::group (C++ function), 409, 940, 1592

sh::settings::Setting::isAdvancedSettings	sh::settings::SettingsManager::_profileNames
(C++ function), 409, 940, 1592	(C++ member), 412, 943, 1595
sh::settings::Setting::isGlobal	sh::settings::SettingsManager::_profileNodes
(C++ function), 409, 941, 1592	(C++ member), 412, 943, 1595
sh::settings::Setting::isPerFileview	sh::settings::SettingsManager::_profileNodesById
(C++ function), 409, 941, 1592	(C++ member), 412, 943, 1595
sh::settings::Setting::name	sh::settings::SettingsManager::_settings
(C++ function), 409, 940, 1592	(C++ member), 412, 943, 1595
sh::settings::Setting::Setting	sh::settings::SettingsManager::_SettingsFinderStru
(C++ function), 409, 940, 1592	(C++ class), 412, 413, 943, 1595
sh::settings::Setting::setValue	sh::settings::SettingsManager::_SettingsFinderStru
(C++ function), 409, 940, 941, 1592	(C++ member), 413, 414, 944, 1596
sh::settings::Setting::valueDescription	sh::settings::SettingsManager::_SettingsFinderStru
(C++ function), 409, 941, 1593	(C++ member), 413, 414, 944, 1596
sh::settings::SettingGroup	sh::settings::SettingsManager::_SettingsFinderStru
(C++ enum), 939, 1590	(C++ member), 413, 414, 944, 1596
sh::settings::SettingGroup::BEHAVIOR	sh::settings::SettingsManager::_SettingsFinderStru
(C++ enumerator), 939, 1591	(C++ member), 413, 414, 944, 1596
sh::settings::SettingGroup::DATAVIEW	sh::settings::SettingsManager::_SettingsFinderStru
(C++ enumerator), 939, 1590	(C++ function), 412, 414, 944, 1595
sh::settings::SettingGroup::FILEHANDLING	sh::settings::SettingsManager::_SettingsFinderStru
(C++ enumerator), 939, 1590	(C++ member), 413, 414, 944, 1596
sh::settings::SettingGroup::GLOBAL	sh::settings::SettingsManager::~SettingsManager
(C++ enumerator), 939, 1590	(C++ function), 410, 942, 1593
sh::settings::SettingGroup::GUI	sh::settings::SettingsManager::applyGlobalSettings
(C++ enumerator), 939, 1590	(C++ function), 410, 941, 1593
sh::settings::SettingGroup::SPECIAL	sh::settings::SettingsManager::applyGlobalSettings
(C++ enumerator), 939, 1591	(C++ function), 410, 941, 1593
sh::settings::SettingsManager	sh::settings::SettingsManager::applyPerEurlSetting
(C++ class), 410, 941, 1593	(C++ function), 410, 941, 1593
sh::settings::SettingsManager::_getSettings	sh::settings::SettingsManager::applyPerEurlSetting
(C++ function), 411, 942, 1594	(C++ function), 410, 941, 1593
sh::settings::SettingsManager::_getSettings	sh::settings::SettingsManager::applyToFileView
(C++ member), 412, 943, 1595	(C++ function), 411, 942, 1594
sh::settings::SettingsManager::_getSettings	sh::settings::SettingsManager::applyToMainWindow
(C++ function), 411, 942, 1594	(C++ function), 411, 942, 1594
sh::settings::SettingsManager::_MyHandle	sh::settings::SettingsManager::doInitialize
(C++ class), 412, 413, 943, 1595	(C++ function), 410, 942, 1593
sh::settings::SettingsManager::_MyHandle	sh::settings::SettingsManager::doShutdown
(C++ member), 412, 413, 943, 1595	(C++ function), 410, 942, 1594
sh::settings::SettingsManager::_MyHandle	sh::settings::SettingsManager::getAllSettings
(C++ function), 412, 413, 943, 1595	(C++ function), 410, 941, 1593
sh::settings::SettingsManager::_MyHandle	sh::settings::SettingsManager::getNodeById
(C++ member), 412, 413, 943, 1595	(C++ function), 410, 941, 1593
sh::settings::SettingsManager::_MyHandle	sh::settings::SettingsManager::getNodesForProfile
(C++ member), 412, 413, 943, 1595	(C++ function), 410, 941, 1593
sh::settings::SettingsManager::_MyHandle	sh::settings::SettingsManager::getProfileList
(C++ member), 412, 413, 943, 1595	(C++ function), 410, 941, 1593
sh::settings::SettingsManager::_MyHandle	sh::settings::SettingsManager::getSettingByName
(C++ function), 412, 413, 943, 1595	(C++ function), 410, 941, 1593
sh::settings::SettingsManager::_MyHandle	sh::settings::SettingsManager::invalidateCache
(C++ function), 412, 413, 943, 1595	(C++ function), 411, 942, 1594
sh::settings::SettingsManager::_MyHandle	sh::settings::SettingsManager::isAlive
(C++ function), 412, 413, 943, 1595	(C++ function), 411, 942, 1594

sh::settings::SettingsManager::profilesChanged (C++ function), 411, 942, 1594  
 sh::settings::SettingsManager::readStoredProfiles (C++ function), 411, 942, 1594  
 sh::settings::SettingsManager::removeNode (C++ function), 410, 941, 1593  
 sh::settings::SettingsManager::removeProfile (C++ function), 410, 941, 1593  
 sh::settings::SettingsManager::SettingsManager (C++ function), 411, 942, 1594  
 sh::settings::SettingsManager::shutdown (C++ function), 411, 942, 1594  
 sh::settings::SettingsManager::storeProfiles (C++ function), 410, 942, 1593  
 sh::settings::SettingsRegistration (C++ class), 414, 944, 1596  
 sh::settings::SettingsRegistration::\_settings (C++ member), 414, 944, 1596  
 sh::settings::SettingsRegistration::~~SettingsRegistration (C++ function), 414, 944, 1596  
 sh::settings::SettingsRegistration::settings (C++ function), 414, 944, 1596  
 sh::settings::SettingsRegistration::SettingsRegistration (C++ function), 414, 944, 1596  
 sh::tools (C++ type), 953, 1613  
 sh::tools::accounts (C++ type), 967, 1627, 1638  
 sh::tools::accounts::AbstractAccountsProvider (C++ class), 441, 967, 1627, 1638  
 sh::tools::accounts::AbstractAccountsProvider::AbstractAccountsProvider (C++ function), 441, 968, 1628, 1639  
 sh::tools::accounts::AbstractAccountsProvider::AbstractAccountsProvider (C++ function), 441, 968, 1628, 1639  
 sh::tools::accounts::AbstractAccountsProvider::AbstractAccountsProvider (C++ function), 441, 968, 1628, 1639  
 sh::tools::accounts::AbstractAccountsProvider::AbstractAccountsProvider (C++ function), 441, 968, 1628, 1639  
 sh::tools::accounts::Account (C++ class), 442, 968, 1628, 1639  
 sh::tools::accounts::Account::Account (C++ function), 442, 968, 1628, 1639  
 sh::tools::accounts::Account::authinfo (C++ member), 442, 968, 1628, 1639  
 sh::tools::accounts::Account::authtype (C++ member), 442, 968, 1628, 1639  
 sh::tools::accounts::Account::domain (C++ member), 442, 968, 1628, 1639  
 sh::tools::accounts::Account::path (C++ member), 442, 968, 1628, 1639  
 sh::tools::accounts::Account::protocol (C++ member), 442, 968, 1628, 1639  
 sh::tools::accounts::Account::server (C++ member), 442, 968, 1628, 1639  
 sh::tools::accounts::Account::serverport (C++ member), 442, 968, 1628, 1639  
 sh::tools::accounts::Account::username (C++ member), 442, 968, 1628, 1639  
 sh::tools::accounts::AccountsManager (C++ class), 442, 968, 1628, 1639  
 sh::tools::accounts::AccountsManager::AccountsManager (C++ function), 443, 969, 1629, 1640  
 sh::tools::accounts::AccountsManager::accountsprovider (C++ member), 443, 969, 1629, 1640  
 sh::tools::accounts::AccountsManager::doInitialize (C++ function), 442, 968, 1628, 1639  
 sh::tools::accounts::AccountsManager::doShutdown (C++ function), 442, 968, 1628, 1639  
 sh::tools::accounts::AccountsManager::findAccounts (C++ function), 442, 968, 1628, 1639  
 sh::tools::accounts::AccountsManager::isAlive (C++ function), 442, 969, 1629, 1640  
 sh::tools::accounts::AccountsManager::mutex (C++ member), 443, 969, 1629, 1640  
 sh::tools::accounts::AccountsManager::registerProvider (C++ function), 442, 968, 1628, 1639  
 sh::tools::accounts::AccountsManager::shutdown (C++ function), 442, 968, 1628, 1639  
 sh::tools::accounts::AccountsManager::storeAccount (C++ function), 442, 968, 1628, 1639  
 sh::tools::accounts::FallbackAccountsProvider (C++ class), 443, 969, 1629, 1640  
 sh::tools::accounts::FallbackAccountsProvider::\_findAccounts (C++ function), 443, 969, 1629, 1640  
 sh::tools::accounts::FallbackAccountsProvider::AbstractAccountsProvider (C++ member), 443, 969, 1629, 1640  
 sh::tools::accounts::FallbackAccountsProvider::doInitialize (C++ function), 443, 969, 1629, 1640  
 sh::tools::accounts::FallbackAccountsProvider::doShutdown (C++ function), 443, 969, 1629, 1640  
 sh::tools::accounts::FallbackAccountsProvider::doStoreAccounts (C++ function), 443, 969, 1629, 1640  
 sh::tools::accounts::FallbackAccountsProvider::findAccounts (C++ function), 443, 969, 1629, 1640  
 sh::tools::accounts::FallbackAccountsProvider::storeAccount (C++ function), 443, 969, 1629, 1640  
 sh::tools::accounts::LibsecretAccountsProvider (C++ class), 443, 969, 1629, 1640  
 sh::tools::accounts::LibsecretAccountsProvider::doInitialize (C++ function), 444, 970, 1630, 1641  
 sh::tools::accounts::LibsecretAccountsProvider::doShutdown (C++ function), 444, 970, 1630, 1641  
 sh::tools::accounts::LibsecretAccountsProvider::findAccounts (C++ function), 444, 970, 1630, 1641  
 sh::tools::accounts::LibsecretAccountsProvider::LibsecretAccountsProvider (C++ function), 444, 970, 1630, 1641  
 sh::tools::accounts::LibsecretAccountsProvider::saveAccount (C++ member), 444, 970, 1630, 1641  
 sh::tools::accounts::LibsecretAccountsProvider::saveAccounts (C++ member), 444, 970, 1630, 1641

sh::tools::accounts::LibsecretAccountsProvider:425,954,1615  
     (C++ *member*), 444, 970, 1630, 1641  
 sh::tools::accounts::LibsecretAccountsProvider:425,954,1615  
     (C++ *member*), 444, 970, 1630, 1641  
 sh::tools::accounts::LibsecretAccountsProvider:425,954,1614  
     (C++ *member*), 444, 970, 1630, 1641  
 sh::tools::accounts::LibsecretAccountsProvider:425,954,1614  
     (C++ *member*), 444, 970, 1630, 1641  
 sh::tools::accounts::LibsecretAccountsProvider:425,954,1614  
     (C++ *function*), 444, 970, 1630, 1641  
 sh::tools::accounts::LibsecretAccountsProvider:954,1614  
     (C++ *member*), 444, 970, 1630, 1641  
 sh::tools::AtomicCounter (C++ *class*), 423,  
     953, 1613  
 sh::tools::AtomicCounter::\_mutex (C++  
     *member*), 424, 953, 1613  
 sh::tools::AtomicCounter::\_value (C++  
     *member*), 424, 953, 1613  
 sh::tools::AtomicCounter::AtomicCounter  
     (C++ *function*), 423, 953, 1613  
 sh::tools::AtomicCounter::doDecValue  
     (C++ *function*), 423, 953, 1613  
 sh::tools::AtomicCounter::doIncValue  
     (C++ *function*), 423, 953, 1613  
 sh::tools::AtomicCounter::Increment  
     (C++ *class*), 424, 953, 1613  
 sh::tools::AtomicCounter::increment  
     (C++ *function*), 423, 953, 1613  
 sh::tools::AtomicCounter::Increment::\_ac  
     (C++ *member*), 424, 953, 1613  
 sh::tools::AtomicCounter::Increment::~Increment(C++  
     *function*), 424, 953, 1613  
 sh::tools::AtomicCounter::Increment::Increment (C++  
     *function*), 424, 953, 1613  
 sh::tools::AtomicCounter::value (C++  
     *function*), 423, 953, 1613  
 sh::tools::Benchmarking (C++ *class*), 424,  
     953, 1614  
 sh::tools::Benchmarking::BenchmarkFrame  
     (C++ *type*), 425, 954, 1614  
 sh::tools::Benchmarking::Benchmarking  
     (C++ *function*), 425, 954, 1614  
 sh::tools::Benchmarking::isAlive (C++  
     *function*), 425, 954, 1614  
 sh::tools::Benchmarking::shutdown (C++  
     *function*), 425, 954, 1614  
 sh::tools::Bookmark (C++ *class*), 425, 954, 1614  
 sh::tools::Bookmark::\_eurl (C++ *member*),  
     425, 954, 1615  
 sh::tools::Bookmark::\_folder (C++ *mem-*  
     *ber*), 425, 954, 1615  
 sh::tools::Bookmark::\_id (C++ *member*), 425,  
     954, 1615  
 sh::tools::Bookmark::\_label (C++ *member*),  
     425, 954, 1615  
 sh::tools::Bookmark::\_tags (C++ *member*),  
     425, 954, 1615  
 sh::tools::Bookmark::Bookmark (C++ *func-*  
     *tion*), 425, 954, 1614  
 sh::tools::Bookmark::eurl (C++ *function*),  
     425, 954, 1614  
 sh::tools::Bookmark::folder (C++ *function*),  
     425, 954, 1614  
 sh::tools::Bookmark::id (C++ *function*), 425,  
     954, 1614  
 sh::tools::Bookmark::label (C++ *function*),  
     425, 954, 1614  
 sh::tools::Bookmark::tags (C++ *function*),  
     425, 954, 1614  
 sh::tools::BookmarkManager (C++ *class*), 426,  
     954, 1615  
 sh::tools::BookmarkManager::\_bookmarks  
     (C++ *member*), 427, 956, 1616  
 sh::tools::BookmarkManager::\_bookmarksmutex  
     (C++ *member*), 427, 956, 1616  
 sh::tools::BookmarkManager::\_changeBookmark  
     (C++ *function*), 427, 955, 1616  
 sh::tools::BookmarkManager::addBookmark  
     (C++ *function*), 426, 955, 1615  
 sh::tools::BookmarkManager::BookmarkManager  
     (C++ *function*), 427, 955, 1616  
 sh::tools::BookmarkManager::cfgvalBookmarks  
     (C++ *member*), 427, 956, 1616  
 sh::tools::BookmarkManager::changeBookmark  
     (C++ *function*), 426, 955, 1615  
 sh::tools::BookmarkManager::changeBookmarkTags  
     (C++ *function*), 426, 955, 1615  
 sh::tools::BookmarkManager::changed  
     (C++ *function*), 427, 955, 1616  
 sh::tools::BookmarkManager::doInitialize  
     (C++ *function*), 426, 955, 1615  
 sh::tools::BookmarkManager::doShutdown  
     (C++ *function*), 426, 955, 1615  
 sh::tools::BookmarkManager::getBookmarks  
     (C++ *function*), 426, 955, 1615  
 sh::tools::BookmarkManager::hasBookmarks  
     (C++ *function*), 426, 955, 1615  
 sh::tools::BookmarkManager::isAlive  
     (C++ *function*), 426, 955, 1615  
 sh::tools::BookmarkManager::moveBookmark  
     (C++ *function*), 427, 955, 1616  
 sh::tools::BookmarkManager::moveBookmarkDown  
     (C++ *function*), 426, 955, 1615  
 sh::tools::BookmarkManager::moveBookmarkToFolder  
     (C++ *function*), 426, 955, 1615  
 sh::tools::BookmarkManager::moveBookmarkUp  
     (C++ *function*), 426, 955, 1615  
 sh::tools::BookmarkManager::readBookmarks



(C++ function), 427, 955, 1616  
 sh::tools::BookmarkManager::removeBookmark (C++ function), 426, 955, 1615  
 sh::tools::BookmarkManager::shutdown (C++ function), 426, 955, 1615  
 sh::tools::BookmarkManager::writeBookmark (C++ function), 427, 955, 1616  
 sh::tools::DataExchange (C++ class), 427, 956, 1616  
 sh::tools::DataExchange::containsFileEntry (C++ function), 427, 956, 1616  
 sh::tools::DataExchange::createCopyAction (C++ function), 427, 956, 1616  
 sh::tools::DataExchange::createMoveAction (C++ function), 427, 956, 1616  
 sh::tools::DataExchange::createPasteAction (C++ function), 428, 956, 1617  
 sh::tools::DataExchange::DataExchange (C++ function), 428, 957, 1617  
 sh::tools::DataExchange::DataExchangeType (C++ enum), 427, 956, 1616  
 sh::tools::DataExchange::DataExchangeType::COPY (C++ enumerator), 427, 956, 1616  
 sh::tools::DataExchange::DataExchangeType::MOVE (C++ enumerator), 427, 956, 1616  
 sh::tools::DataExchange::doInitialize (C++ function), 428, 956, 1617  
 sh::tools::DataExchange::doShutdown (C++ function), 428, 957, 1617  
 sh::tools::DataExchange::FilelistTypeGnome (C++ member), 428, 957, 1617  
 sh::tools::DataExchange::FilelistTypePlasma (C++ member), 428, 957, 1617  
 sh::tools::DataExchange::FilelistTypeShashot (C++ member), 428, 957, 1617  
 sh::tools::DataExchange::FilelistTypeUri (C++ member), 428, 957, 1617  
 sh::tools::DataExchange::getMimeDataFromFilesystem (C++ function), 427, 956, 1616  
 sh::tools::DataExchange::getSources (C++ function), 428, 956, 1617  
 sh::tools::DataExchange::isAlive (C++ function), 428, 957, 1617  
 sh::tools::DataExchange::linebreak (C++ member), 428, 957, 1617  
 sh::tools::DataExchange::shutdown (C++ function), 428, 957, 1617  
 sh::tools::filetypes (C++ type), 970, 1630, 1641  
 sh::tools::filetypes::FileTypeManager (C++ class), 444, 970, 1630, 1641  
 sh::tools::filetypes::FileTypeManager::\_mutex (C++ member), 445, 971, 1631, 1642  
 sh::tools::filetypes::FileTypeManager::\_openMethod (C++ member), 445, 971, 1631, 1642  
 sh::tools::filetypes::FileTypeManager::determineMimeType (C++ function), 444, 970, 1630, 1642  
 sh::tools::filetypes::FileTypeManager::doInitialize (C++ function), 445, 971, 1631, 1642  
 sh::tools::filetypes::FileTypeManager::doShutdown (C++ function), 445, 971, 1631, 1642  
 sh::tools::filetypes::FileTypeManager::FileTypeManager (C++ function), 445, 971, 1631, 1642  
 sh::tools::filetypes::FileTypeManager::getAllOpenMethods (C++ function), 445, 971, 1631, 1642  
 sh::tools::filetypes::FileTypeManager::getOpenMethod (C++ function), 444, 970, 1630, 1642  
 sh::tools::filetypes::FileTypeManager::isAlive (C++ function), 445, 971, 1631, 1642  
 sh::tools::filetypes::FileTypeManager::MimeTypeDetector (C++ class), 445, 446, 971, 1631, 1642  
 sh::tools::filetypes::FileTypeManager::MimeTypeDetector::COPY (C++ function), 445, 446, 971, 1631, 1643  
 sh::tools::filetypes::FileTypeManager::MimeTypeDetector::MOVE (C++ function), 445, 446, 971, 1631, 1643  
 sh::tools::filetypes::FileTypeManager::MimeTypeInfo (C++ class), 445, 446, 971, 1631, 1643  
 sh::tools::filetypes::FileTypeManager::MimeTypeInfo::Gnome (C++ function), 446, 447, 972, 1632, 1643  
 sh::tools::filetypes::FileTypeManager::MimeTypeInfo::Plasma (C++ function), 446, 447, 972, 1632, 1643  
 sh::tools::filetypes::FileTypeManager::MimeTypeInfo::Shashot (C++ function), 446, 447, 972, 1632, 1643  
 sh::tools::filetypes::FileTypeManager::MimeTypeInfo::Uri (C++ function), 446, 447, 972, 1632, 1643  
 sh::tools::filetypes::FileTypeManager::shutdown (C++ function), 445, 971, 1631, 1642  
 sh::tools::filetypes::FreeDesktopOrgMimeTypeInformation (C++ class), 447, 972, 1632, 1643  
 sh::tools::filetypes::FreeDesktopOrgMimeTypeInformation::Gnome (C++ member), 447, 972, 1632, 1643  
 sh::tools::filetypes::FreeDesktopOrgMimeTypeInformation::Plasma (C++ member), 447, 972, 1632, 1643  
 sh::tools::filetypes::FreeDesktopOrgMimeTypeInformation::Shashot (C++ member), 447, 972, 1632, 1643  
 sh::tools::filetypes::FreeDesktopOrgMimeTypeInformation::Uri (C++ member), 447, 972, 1632, 1643  
 sh::tools::filetypes::FreeDesktopOrgMimeTypeInformation::MimeTypeDetector (C++ function), 447, 972, 1632, 1643  
 sh::tools::filetypes::FreeDesktopOrgMimeTypeInformation::MimeTypeInfo (C++ function), 447, 972, 1632, 1643



(C++ function), 451, 976, 1636, 1647  
 sh::tools::filetypes::UserDefinedOpenMethodDetector (C++ function), 421, 959, 1619  
 (C++ function), 451, 976, 1636, 1647  
 sh::tools::filetypes::UserDefinedOpenMethodDetector (C++ function), 421, 959, 1619  
 (C++ function), 451, 976, 1636, 1647  
 sh::tools::filetypes::UserDefinedOpenMethodDetector (C++ function), 421, 959, 1619  
 (C++ function), 451, 976, 1636, 1647  
 sh::tools::HistoryTracker (C++ class), 428, 957, 1617  
 sh::tools::HistoryTracker::\_current (C++ member), 429, 958, 1618  
 sh::tools::HistoryTracker::\_items (C++ member), 429, 958, 1618  
 sh::tools::HistoryTracker::backwardList (C++ function), 429, 957, 1618  
 sh::tools::HistoryTracker::count (C++ function), 429, 957, 1618  
 sh::tools::HistoryTracker::forwardList (C++ function), 429, 957, 1618  
 sh::tools::HistoryTracker::HISTORY\_SIZE (C++ member), 429, 958, 1618  
 sh::tools::HistoryTracker::HistoryEntry (C++ class), 429, 958, 1618  
 sh::tools::HistoryTracker::HistoryEntry::\_index (C++ member), 429, 430, 958, 1618  
 sh::tools::HistoryTracker::HistoryEntry::\_value (C++ member), 429, 430, 958, 1618  
 sh::tools::HistoryTracker::HistoryEntry::HistoryEntry (C++ class), 432, 433, 960, 1620  
 (C++ function), 429, 958, 1618  
 sh::tools::HistoryTracker::HistoryEntry::index (C++ member), 432, 960, 1620  
 (C++ function), 429, 958, 1618  
 sh::tools::HistoryTracker::HistoryEntry::value (C++ function), 432, 433, 960, 1620  
 (C++ function), 429, 958, 1618  
 sh::tools::HistoryTracker::HistoryTracker (C++ function), 429, 957, 1618  
 sh::tools::HistoryTracker::revisitValue (C++ function), 429, 957, 1618  
 sh::tools::HistoryTracker::visitValue (C++ function), 429, 957, 1618  
 sh::tools::Jsonable (C++ class), 430, 958, 1618  
 sh::tools::Jsonable::toJson (C++ function), 430, 958, 1619  
 sh::tools::LocalFile (C++ class), 430, 958, 1619  
 sh::tools::LocalFile::\_istemp (C++ member), 430, 959, 1619  
 sh::tools::LocalFile::isTemp (C++ function), 430, 958, 1619  
 sh::tools::LocalFile::LocalFile (C++ function), 430, 958, 1619  
 sh::tools::LocalFilesystemWatcher (C++ class), 430, 959, 1619  
 sh::tools::LocalFilesystemWatcher::~~LocalFilesystemWatcher (C++ function), 431, 959, 1619  
 sh::tools::LocalFilesystemWatcher::addDirectory (C++ function), 431, 959, 1619  
 sh::tools::LocalFilesystemWatcher::addFile (C++ function), 431, 959, 1619  
 sh::tools::LocalFilesystemWatcher::doCustomOpenMethod (C++ function), 431, 959, 1619  
 sh::tools::LocalFilesystemWatcher::doInitialize (C++ function), 431, 959, 1619  
 sh::tools::LocalFilesystemWatcher::doShutdown (C++ function), 431, 959, 1619  
 sh::tools::LocalFilesystemWatcher::elementCreated (C++ function), 431, 959, 1620  
 sh::tools::LocalFilesystemWatcher::elementDeleted (C++ function), 431, 959, 1620  
 sh::tools::LocalFilesystemWatcher::elementModified (C++ function), 431, 959, 1620  
 sh::tools::LocalFilesystemWatcher::emitElementCreated (C++ function), 431, 960, 1620  
 sh::tools::LocalFilesystemWatcher::emitElementDeleted (C++ function), 431, 960, 1620  
 sh::tools::LocalFilesystemWatcher::emitElementModified (C++ function), 431, 960, 1620  
 sh::tools::LocalFilesystemWatcher::id2inotifywd (C++ member), 432, 960, 1620  
 sh::tools::LocalFilesystemWatcher::id2path (C++ member), 432, 960, 1620  
 sh::tools::LocalFilesystemWatcher::inotifymutex (C++ member), 432, 960, 1620  
 sh::tools::LocalFilesystemWatcher::InotifyThread (C++ class), 432, 433, 960, 1620  
 sh::tools::LocalFilesystemWatcher::inotifyThread (C++ member), 432, 960, 1620  
 sh::tools::LocalFilesystemWatcher::InotifyThread::InotifyThread (C++ member), 432, 960, 1620  
 sh::tools::LocalFilesystemWatcher::inotifythreadmutex (C++ member), 432, 960, 1620  
 sh::tools::LocalFilesystemWatcher::inotifythreadmutex (C++ member), 432, 960, 1620  
 sh::tools::LocalFilesystemWatcher::inotifywd2ids (C++ member), 432, 960, 1620  
 sh::tools::LocalFilesystemWatcher::isAlive (C++ function), 431, 959, 1619  
 sh::tools::LocalFilesystemWatcher::LocalFilesystemWatcher (C++ function), 431, 960, 1620  
 sh::tools::LocalFilesystemWatcher::nextEid (C++ member), 432, 960, 1620  
 sh::tools::LocalFilesystemWatcher::removeDirectory (C++ function), 431, 959, 1619  
 sh::tools::LocalFilesystemWatcher::removeFile (C++ function), 431, 959, 1619  
 sh::tools::LocalFilesystemWatcher::shutdown (C++ function), 431, 959, 1619  
 sh::tools::LocalFilesystemWatcherConnector (C++ class), 432, 960, 1620  
 sh::tools::LocalFilesystemWatcherConnector::dir2watcher (C++ member), 433, 961, 1621

sh::tools::LocalFilesystemWatcherConnector::doInit (C++ function), 434, 962, 1622  
 (C++ function), 432, 960, 1621 sh::tools::OperationsCache::getOperationForContain  
 sh::tools::LocalFilesystemWatcherConnector::doShutdown (C++ function), 434, 962, 1622  
 (C++ function), 432, 960, 1621 sh::tools::OperationsCache::isAlive  
 sh::tools::LocalFilesystemWatcherConnector::get (C++ function), 435, 962, 1622  
 (C++ function), 433, 961, 1621 sh::tools::OperationsCache::OperationsCache  
 sh::tools::LocalFilesystemWatcherConnector::isAlive (C++ function), 435, 963, 1623  
 (C++ function), 432, 960, 1621 sh::tools::OperationsCache::shutdown  
 sh::tools::LocalFilesystemWatcherConnector::Load (C++ function), 434, 962, 1622  
 (C++ function), 433, 961, 1621 sh::tools::ReadDataDevice (C++ class), 435,  
 sh::tools::LocalFilesystemWatcherConnector::shutdown 963, 1623  
 (C++ function), 432, 960, 1621 sh::tools::ReadDataDevice::current (C++  
 sh::tools::LocalFilesystemWatcherConnector::wait (C++ member), 435, 963, 1623  
 (C++ member), 433, 961, 1621 sh::tools::ReadDataDevice::currentconsumed  
 sh::tools::Misc (C++ class), 433, 961, 1621 (C++ member), 435, 963, 1623  
 sh::tools::Misc::\_hash (C++ function), 434, 962, 1622 sh::tools::ReadDataDevice::currentlen  
 sh::tools::Misc::\_jundefined (C++ member), 434, 962, 1622 (C++ member), 435, 963, 1623  
 sh::tools::Misc::\_qnetwork (C++ member), 434, 962, 1622 sh::tools::ReadDataDevice::getdata (C++  
 sh::tools::Misc::compareHash (C++ function), 433, 961, 1621 function), 435, 963, 1623  
 sh::tools::Misc::generateUniqueHash sh::tools::ReadDataDevice::ReadDataDevice  
 (C++ function), 434, 961, 1622 (C++ function), 435, 963, 1623  
 sh::tools::Misc::hash (C++ function), 433, 961, 1621 sh::tools::ThumbnailManager (C++ class),  
 sh::tools::Misc::hashUnsalted (C++ function), 433, 961, 1621 435, 963, 1623  
 sh::tools::Misc::iconToBase64SrcEncodings sh::tools::ThumbnailManager::\_curr\_accessTime  
 (C++ function), 433, 961, 1621 (C++ member), 437, 964, 1624  
 sh::tools::Misc::iconToPngByteArray sh::tools::ThumbnailManager::\_enforce\_capacity  
 (C++ function), 433, 961, 1621 (C++ function), 437, 964, 1624  
 sh::tools::Misc::jsonableToJson (C++ function), 434, 961, 1621 sh::tools::ThumbnailManager::\_thumbnailProviderMap  
 sh::tools::Misc::makeHttpRequest (C++ function), 434, 961, 1622 (C++ member), 437, 965, 1625  
 sh::tools::Misc::Misc (C++ function), 434, 962, 1622 sh::tools::ThumbnailManager::\_thumbnailProviders  
 sh::tools::Misc::qjsonToJson (C++ function), 433, 961, 1621 (C++ member), 437, 965, 1625  
 sh::tools::Misc::qmapToJson (C++ function), 434, 961, 1621 sh::tools::ThumbnailManager::\_thumbnailProvidersMut  
 sh::tools::Misc::randomBytes (C++ function), 434, 961, 1622 (C++ member), 437, 965, 1625  
 sh::tools::OperationsCache (C++ class), 434, 962, 1622 sh::tools::ThumbnailManager::addThumbnailProvider  
 sh::tools::OperationsCache::cache (C++ member), 435, 963, 1623 (C++ function), 436, 963, 1623  
 sh::tools::OperationsCache::cachemutex sh::tools::ThumbnailManager::capacity  
 (C++ member), 435, 963, 1623 (C++ member), 437, 964, 1624  
 sh::tools::OperationsCache::doInitializes sh::tools::ThumbnailManager::doInitialize  
 (C++ function), 434, 962, 1622 (C++ function), 436, 963, 1623  
 sh::tools::OperationsCache::doShutdown sh::tools::ThumbnailManager::getThumbnail  
 sh::tools::ThumbnailManager::isAlive  
 (C++ function), 436, 964, 1624  
 sh::tools::ThumbnailManager::maxworkers  
 (C++ member), 437, 964, 1624  
 sh::tools::ThumbnailManager::REGISTER\_THUMBNAI  
 (C++ member), 436, 964, 1624  
 sh::tools::ThumbnailManager::REGISTER\_THUMBNAI  
 (C++ member), 436, 964, 1624  
 sh::tools::ThumbnailManager::REGISTER\_THUMBNAI  
 (C++ member), 436, 964, 1624  
 sh::tools::ThumbnailManager::REGISTER\_THUMBNAI  
 (C++ member), 436, 964, 1624





```

(C++ function), 453, 977, 1637, 1649
sh::tools::thumbnailproviders::ImageMagickThumbnailGenerator (C++ class), 440, 966, 1626
(C++ member), 453, 978, 1638, 1649
sh::tools::thumbnailproviders::PlaintextThumbnailProvider (C++ class), 453, 978, 1638, 1649
(C++ class), 453, 978, 1638, 1649
sh::tools::thumbnailproviders::PlaintextThumbnailProvider (C++ function), 487, 979, 1651
(C++ member), 454, 978, 1638, 1650
sh::tools::thumbnailproviders::PlaintextThumbnailProvider (C++ function), 487, 979, 1651
(C++ function), 454, 978, 1638, 1650
sh::tools::thumbnailproviders::PlaintextThumbnailProvider (C++ function), 487, 979, 1651
(C++ function), 454, 978, 1638, 1650
sh::tools::thumbnailproviders::PlaintextThumbnailProvider (C++ function), 487, 979, 1651
(C++ function), 453, 978, 1638, 1649
sh::tools::thumbnailproviders::PlaintextThumbnailProvider (C++ class), 487, 979, 1651
(C++ function), 453, 978, 1638, 1649
sh::tools::UserDirLock (C++ class), 440, 966, 1626
sh::tools::UserDirLock::~UserDirLock (C++ function), 440, 966, 1626
sh::tools::UserDirLock::currentThread (C++ member), 440, 966, 1626
sh::tools::UserDirLock::dounlock (C++ member), 440, 966, 1626
sh::tools::UserDirLock::mutex (C++ member), 440, 966, 1626
sh::tools::UserDirLock::slockfile (C++ member), 440, 966, 1626
sh::tools::UserDirLock::slockpref (C++ member), 440, 966, 1626
sh::tools::UserDirLock::UserDirLock (C++ function), 440, 966, 1626
sh::tools::VisibleViews (C++ class), 440, 966, 1626
sh::tools::VisibleViews::doInitialize (C++ function), 440, 967, 1627
sh::tools::VisibleViews::doShutdown (C++ function), 441, 967, 1627
sh::tools::VisibleViews::isAlive (C++ function), 441, 967, 1627
sh::tools::VisibleViews::onenteredhandlers (C++ member), 441, 967, 1627
sh::tools::VisibleViews::onlefthandlers (C++ member), 441, 967, 1627
sh::tools::VisibleViews::registerOnEnteredHandlers (C++ function), 440, 967, 1627
sh::tools::VisibleViews::registerOnLeftHandlers (C++ function), 440, 967, 1627
sh::tools::VisibleViews::shutdown (C++ function), 441, 967, 1627
sh::tools::VisibleViews::viewEnteredDirectories (C++ function), 440, 967, 1627
sh::tools::VisibleViews::visibleDirs (C++ member), 441, 967, 1627
sh::tools::VisibleViews::VisibleViews (C++ function), 440, 967, 1627
(C++ function), 441, 967, 1627
sh::ui::ActionExecutionInfoPanel_HelperQObject (C++ class), 486, 979, 1650
sh::ui::ActionExecutionInfoPanel_HelperQObject::ActionExecutionInfoPanel_HelperQObject (C++ function), 487, 979, 1651
sh::ui::ActionExecutionInfoPanel_HelperQObject::close (C++ function), 487, 979, 1651
sh::ui::ActionExecutionInfoPanel_HelperQObject::currentDirectory (C++ function), 487, 979, 1651
sh::ui::ActionExecutionInfoPanel_HelperQObject::currentFileViewOptions (C++ function), 487, 979, 1651
sh::ui::ActionExecutionInfoPanel_HelperQObject::currentProfileChosen (C++ function), 487, 979, 1651
sh::ui::ActionExecutionInfoPanel_HelperQObject::fileViewCountChanged (C++ function), 487, 979, 1651
sh::ui::ActionExecutionInfoPanel_HelperQObject::fileViewOptionsChanged (C++ function), 487, 979, 1651
sh::ui::ActionExecutionInfoPanel_HelperQObject::globalViewOptionsChanged (C++ function), 487, 979, 1651
sh::ui::ActionExecutionInfoPanel_HelperQObject::searchPanelConfiguration (C++ class), 488, 980, 1651
sh::ui::AboutDialog (C++ class), 454, 980, 1651
sh::ui::AboutDialog::AboutDialog (C++ function), 454, 980, 1652
sh::ui::AboutDialog::close (C++ function), 454, 980, 1652
sh::ui::AboutDialog::dialogId (C++ function), 454, 980, 1652
sh::ui::AboutDialog::isInitiated (C++ function), 454, 980, 1652

```

tion), 454, 980, 1652

sh::ui::AboutDialog::manager (C++ function), 454, 981, 1652

sh::ui::AboutDialog::waitClosed (C++ function), 454, 980, 1652

sh::ui::AboutDialog::wasAccepted (C++ function), 454, 980, 1652

sh::ui::AboutDialog::wasClosed (C++ function), 454, 981, 1652

sh::ui::ActionExecutionInfoDialog (C++ class), 455, 981, 1652

sh::ui::ActionExecutionInfoDialog::ActionExecutionInfoDialog (C++ function), 455, 981, 1653

sh::ui::ActionExecutionInfoDialog::credentialsDialog (C++ function), 456, 982, 1653

sh::ui::ActionExecutionInfoDialog::inputBox (C++ function), 455, 982, 1653

sh::ui::ActionExecutionInfoDialog::isBackground (C++ function), 455, 981, 1653

sh::ui::ActionExecutionInfoDialog::isLogicallyVisible (C++ function), 455, 981, 1653

sh::ui::ActionExecutionInfoDialog::messageBox (C++ function), 455, 982, 1653

sh::ui::ActionExecutionInfoDialog::MessageBoxButton (C++ enum), 455, 981, 1652

sh::ui::ActionExecutionInfoDialog::MessageBoxButtonActionExecutionInfoPanel (C++ enumerator), 455, 981, 1652

sh::ui::ActionExecutionInfoDialog::MessageBoxButtonColumnDimensions (C++ enumerator), 455, 981, 1652

sh::ui::ActionExecutionInfoDialog::MessageBoxButtonColumnDimensions::ColumnDimensions (C++ enumerator), 455, 981, 1653

sh::ui::ActionExecutionInfoDialog::MessageBoxButtonColumnDimensions::db (C++ function), 457, 983, 1655

sh::ui::ActionExecutionInfoDialog::MessageBoxButtonColumnDimensions::dims (C++ member), 457, 983, 1655

sh::ui::ActionExecutionInfoDialog::MessageBoxButtonColumnDimensions::getWidth (C++ function), 457, 983, 1654

sh::ui::ActionExecutionInfoDialog::MessageBoxButtonColumnDimensions::index (C++ member), 457, 983, 1655

sh::ui::ActionExecutionInfoDialog::multilineInputBoxColumnDimensions::remove (C++ function), 457, 983, 1654

sh::ui::ActionExecutionInfoDialog::setBackgroundColumnDimensions::setWidth (C++ function), 457, 983, 1654

sh::ui::ActionExecutionInfoDialog::setDetails (C++ function), 455, 981, 1653

sh::ui::ActionExecutionInfoDialog::setForeground (C++ class), 457, 983, 1655

sh::ui::ActionExecutionInfoDialog::setHead (C++ function), 455, 981, 1653

sh::ui::ActionExecutionInfoDialog::setLogicallyVisible (C++ function), 455, 981, 1653

sh::ui::ActionExecutionInfoDialog::setProgress (C++ function), 455, 981, 1653

sh::ui::ActionExecutionInfoDialog::simpleChooseForm (C++ function), 456, 982, 1653

sh::ui::ActionExecutionInfoDialog::simpleInputDialog (C++ function), 456, 982, 1653

sh::ui::ActionExecutionInfoDialog::simpleMessageBox (C++ function), 456, 982, 1653

sh::ui::ActionExecutionInfoDialog::unixPermissionsDialog (C++ function), 456, 982, 1653

sh::ui::ActionExecutionInfoPanel (C++ class), 456, 982, 1654

sh::ui::ActionExecutionInfoPanel::~~ActionExecutionInfoPanel (C++ function), 456, 983, 1654

sh::ui::ActionExecutionInfoPanel::onClicked (C++ function), 456, 982, 1654

sh::ui::ActionExecutionInfoPanel::onDestroyed (C++ function), 456, 983, 1654

sh::ui::ActionExecutionInfoPanel::onVisibilityChange (C++ function), 456, 983, 1654

sh::ui::ActionExecutionInfoPanel::setForceForeground (C++ function), 456, 982, 1654

sh::ui::ActionExecutionInfoPanel::setLabel (C++ function), 456, 982, 1654

sh::ui::ActionExecutionInfoPanel::setPanelVisible (C++ function), 456, 982, 1654

sh::ui::ActionExecutionInfoPanel::setProgress (C++ function), 456, 982, 1654

sh::ui::ActionExecutionInfoPanel::setWidth (C++ function), 456, 982, 1654

sh::ui::Dialog::~~Dialog (C++ *function*), 458, 984, 1655  
 sh::ui::Dialog::~close (C++ *function*), 458, 984, 1656  
 sh::ui::Dialog::~Dialog (C++ *function*), 458, 984, 1655  
 sh::ui::Dialog::~dialogId (C++ *function*), 458, 984, 1655  
 sh::ui::Dialog::~isInited (C++ *function*), 458, 984, 1655  
 sh::ui::Dialog::~manager (C++ *function*), 458, 984, 1656  
 sh::ui::Dialog::~setup (C++ *function*), 458, 984, 1656  
 sh::ui::Dialog::~waitClosed (C++ *function*), 458, 984, 1655  
 sh::ui::Dialog::~wasAccepted (C++ *function*), 458, 984, 1655  
 sh::ui::Dialog::~wasClosed (C++ *function*), 458, 984, 1656  
 sh::ui::DialogManager (C++ *class*), 459, 985, 1656  
 sh::ui::DialogManager::\_nextDialogId (C++ *member*), 460, 986, 1658  
 sh::ui::DialogManager::\_openDialogs (C++ *member*), 460, 986, 1658  
 sh::ui::DialogManager::~~DialogManager (C++ *function*), 459, 985, 1657  
 sh::ui::DialogManager::closeDialog (C++ *function*), 460, 986, 1657  
 sh::ui::DialogManager::createAndShowDialog (C++ *function*), 459, 985, 1657  
 sh::ui::DialogManager::getAllDialogIds (C++ *function*), 459, 985, 1657  
 sh::ui::DialogManager::getAllDialogs (C++ *function*), 459, 985, 1657  
 sh::ui::DialogManager::getDialogById (C++ *function*), 459, 985, 1657  
 sh::ui::DialogManager::initAndShowDialog (C++ *function*), 460, 986, 1657  
 sh::ui::DialogManager::showDialog (C++ *function*), 460, 986, 1657  
 sh::ui::DialogManager::stopDialog (C++ *function*), 460, 986, 1657  
 sh::ui::DialogManager::waitClosed (C++ *function*), 460, 986, 1657  
 sh::ui::DialogManager::wasAccepted (C++ *function*), 460, 986, 1657  
 sh::ui::ExceptionDialog (C++ *class*), 460, 986, 1658  
 sh::ui::ExceptionDialog::\_details (C++ *member*), 462, 988, 1659  
 sh::ui::ExceptionDialog::\_error1 (C++ *member*), 462, 988, 1659  
 sh::ui::ExceptionDialog::\_error2 (C++ *member*), 462, 988, 1659  
 sh::ui::ExceptionDialog::\_icon (C++ *member*), 462, 988, 1659  
 sh::ui::ExceptionDialog::\_mayCancel (C++ *member*), 462, 988, 1659  
 sh::ui::ExceptionDialog::\_mayClose (C++ *member*), 462, 988, 1659  
 sh::ui::ExceptionDialog::\_mayRetry (C++ *member*), 462, 988, 1659  
 sh::ui::ExceptionDialog::\_showLoglabelAndDetails (C++ *member*), 462, 988, 1659  
 sh::ui::ExceptionDialog::answer (C++ *function*), 461, 987, 1658  
 sh::ui::ExceptionDialog::close (C++ *function*), 462, 987, 1659  
 sh::ui::ExceptionDialog::details (C++ *function*), 461, 987, 1658  
 sh::ui::ExceptionDialog::dialogId (C++ *function*), 461, 987, 1659  
 sh::ui::ExceptionDialog::error1 (C++ *function*), 461, 987, 1658  
 sh::ui::ExceptionDialog::error2 (C++ *function*), 461, 987, 1658  
 sh::ui::ExceptionDialog::ExceptionDialog (C++ *function*), 461, 986, 1658  
 sh::ui::ExceptionDialog::icon (C++ *function*), 461, 987, 1658  
 sh::ui::ExceptionDialog::isInited (C++ *function*), 461, 987, 1659  
 sh::ui::ExceptionDialog::manager (C++ *function*), 462, 987, 1659  
 sh::ui::ExceptionDialog::mayCancel (C++ *function*), 461, 987, 1658  
 sh::ui::ExceptionDialog::mayClose (C++ *function*), 461, 987, 1658  
 sh::ui::ExceptionDialog::mayRetry (C++ *function*), 461, 987, 1658  
 sh::ui::ExceptionDialog::showLoglabelAndDetails (C++ *function*), 461, 987, 1658  
 sh::ui::ExceptionDialog::waitClosed (C++ *function*), 461, 987, 1659  
 sh::ui::ExceptionDialog::wasAccepted (C++ *function*), 461, 987, 1659  
 sh::ui::ExceptionDialog::wasClosed (C++ *function*), 462, 987, 1659  
 sh::ui::ExceptionDialogResult (C++ *enum*), 979, 1650  
 sh::ui::ExceptionDialogResult::Cancel (C++ *enumerator*), 979, 1650  
 sh::ui::ExceptionDialogResult::Close (C++ *enumerator*), 979, 1650  
 sh::ui::ExceptionDialogResult::Retry (C++ *enumerator*), 979, 1650



sh::ui::ExceptionDialogResult::Shutdown (C++ enumerator), 979, 1650  
 sh::ui::FilePropertyDialog (C++ class), 462, 988, 1659  
 sh::ui::FilePropertyDialog::\_nodes (C++ member), 464, 989, 1661  
 sh::ui::FilePropertyDialog::\_propertytabsh::ui::FilePropertyDialogTab::title (C++ member), 464, 989, 1661  
 sh::ui::FilePropertyDialog::\_tabs (C++ member), 464, 989, 1661  
 sh::ui::FilePropertyDialog::~FilePropertyDialog (C++ function), 462, 988, 1660  
 sh::ui::FilePropertyDialog::addTabFactorysh::ui::FilePropertyDialogTab::widgetAt (C++ function), 463, 989, 1661  
 sh::ui::FilePropertyDialog::close (C++ function), 463, 989, 1660  
 sh::ui::FilePropertyDialog::createTabViewsh::ui::FilePropertyDialogTabActionsView (C++ function), 463, 988, 1660  
 sh::ui::FilePropertyDialog::createTabViewsh::ui::FilePropertyDialogTabActionsView::buttons (C++ function), 462, 988, 1660  
 sh::ui::FilePropertyDialog::createTabViewsh::ui::FilePropertyDialogTabActionsView::content (C++ function), 463, 988, 1660  
 sh::ui::FilePropertyDialog::dialogId sh::ui::FilePropertyDialogTabActionsView::FilePropertyDialog (C++ function), 463, 988, 1660  
 sh::ui::FilePropertyDialog::FilePropertyDialogsh::ui::FilePropertyDialogTabActionsView::onButton (C++ function), 462, 988, 1660  
 sh::ui::FilePropertyDialog::isInitied sh::ui::FilePropertyDialogTabActionsView::setButton (C++ function), 463, 988, 1660  
 sh::ui::FilePropertyDialog::manager sh::ui::FilePropertyDialogTabActionsView::setCenter (C++ function), 463, 989, 1660  
 sh::ui::FilePropertyDialog::refresh sh::ui::FilePropertyDialogTabActionsView::setVisible (C++ function), 462, 988, 1660  
 sh::ui::FilePropertyDialog::tabs (C++ function), 462, 988, 1660  
 sh::ui::FilePropertyDialog::waitClosed sh::ui::FilePropertyDialogTabFactory (C++ function), 463, 989, 1660  
 sh::ui::FilePropertyDialog::wasAccepted sh::ui::FilePropertyDialogTabFactory::~FilePropertyDialog (C++ function), 463, 988, 1660  
 sh::ui::FilePropertyDialog::wasClosed sh::ui::FilePropertyDialogTabFactory::construct (C++ function), 463, 989, 1660  
 sh::ui::FilePropertyDialog::widgetAt sh::ui::FilePropertyDialogTabFactory::REGISTER\_FILE (C++ function), 462, 989, 1660  
 sh::ui::FilePropertyDialog::widgetCount sh::ui::FilePropertyDialogTabFactory::REGISTER\_FILE (C++ function), 462, 988, 1660  
 sh::ui::FilePropertyDialogTab (C++ class), 464, 989, 1661  
 sh::ui::FilePropertyDialogTab::\_dialog sh::ui::FilePropertyDialogTabFactory::REGISTER\_FILE (C++ member), 465, 990, 1662  
 sh::ui::FilePropertyDialogTab::~FilePropertyDialogsh::ui::FilePropertyDialogTabFactoryByFunction (C++ function), 465, 990, 1662  
 sh::ui::FilePropertyDialogTab::dialog sh::ui::FilePropertyDialogTabFactoryByFunction::con (C++ function), 465, 990, 1662  
 sh::ui::FilePropertyDialogTab::nodes sh::ui::FilePropertyDialogTabFactoryByFunction::Fil (C++ function), 464, 990, 1662

sh::ui::FilePropertyDialogTabFactoryByFunction: *function*, 458, 994, 1666  
 (C++ member), 467, 992, 1664  
 sh::ui::FilePropertyDialogTabFactoryByFunction: *function*, 470, 995, 1667  
 (C++ member), 467, 993, 1664  
 sh::ui::FilePropertyDialogTabFactoryByFunction: *function*, 469, 994, 1666  
 (C++ member), 467, 992, 1664  
 sh::ui::FilePropertyDialogTabFactoryByFunction: *function*, 470, 995, 1667  
 (C++ member), 467, 992, 1664  
 sh::ui::FilePropertyDialogTabFactoryByFunction: *function*, 469, 994, 1666  
 (C++ member), 467, 992, 1664  
 sh::ui::FilePropertyDialogTabFactoryByFunction: *function*, 470, 995, 1667  
 (C++ member), 467, 992, 1664  
 sh::ui::FilePropertyDialogTabIconTextBannerView: 469, 994, 1666  
 (C++ class), 467, 993, 1664  
 sh::ui::FilePropertyDialogTabIconTextBannerView: *function*, 470, 995, 1666  
 (C++ function), 468, 993, 1664  
 sh::ui::FilePropertyDialogTabIconTextBannerView: *function*, 469, 994, 1666  
 (C++ function), 468, 993, 1664  
 sh::ui::FilePropertyDialogTabIconTextBannerView: *function*, 470, 995, 1666  
 (C++ function), 468, 993, 1664  
 sh::ui::FilePropertyDialogTabIconTextBannerView: *function*, 469, 994, 1666  
 (C++ function), 468, 993, 1664  
 sh::ui::FilePropertyDialogTabIconTextBannerView: *function*, 470, 995, 1667  
 (C++ function), 468, 993, 1664  
 sh::ui::FilePropertyDialogTabTableView: 994, 1666  
 (C++ class), 468, 993, 1664  
 sh::ui::FilePropertyDialogTabTableView::FilePropertyDialogTabTableView: 995, 1667  
 (C++ function), 468, 993, 1665  
 sh::ui::FilePropertyDialogTabTableView::getItem: *function*, 470, 995, 1666  
 (C++ function), 468, 993, 1665  
 sh::ui::FilePropertyDialogTabTableView::getSelectedItem: *function*, 470, 995, 1667  
 (C++ function), 468, 993, 1665  
 sh::ui::FilePropertyDialogTabTableView::ItemIndex: *function*, 470, 995, 1667  
 (C++ type), 468, 993, 1665  
 sh::ui::FilePropertyDialogTabTableView::setContent: *function*, 470, 995, 1666  
 (C++ function), 468, 993, 1665  
 sh::ui::FilePropertyDialogTabTextView: *function*, 470, 995, 1666  
 (C++ class), 468, 993, 1665  
 sh::ui::FilePropertyDialogTabTextView::FilePropertyDialogTabTextView: 470, 995, 1666  
 (C++ function), 469, 994, 1665  
 sh::ui::FilePropertyDialogTabTextView::setContent: *function*, 470, 995, 1666  
 (C++ function), 469, 994, 1665  
 sh::ui::FilePropertyDialogTabViewContent: 470, 995, 1666  
 (C++ class), 469, 994, 1665  
 sh::ui::FilePropertyDialogTabViewContent::~FilePropertyDialogTabViewContent: 469, 994, 1666  
 (C++ function), 469, 994, 1665  
 sh::ui::FilePropertyDialogTabViewContent::FilePropertyDialogTabViewContent: 470, 995, 1666  
 (C++ function), 469, 994, 1665  
 sh::ui::FileView (C++ class), 469, 994, 1665  
 sh::ui::FileView::\_model (C++ member), 471, 996, 1667  
 sh::ui::FileView::\_node (C++ member), 471, 996, 1667  
 sh::ui::FileView::\_visibleviewslifetimecallback: *function*, 471, 996, 1667  
 (C++ member), 471, 996, 1667  
 sh::ui::FileView::~FileView (C++ function), 469, 994, 1666  
 sh::ui::FileView::columnDimensions (C++

sh::ui::FileView::filemodel (C++ function),  
 sh::ui::FileView::FileView (C++ function),  
 sh::ui::FileView::getAllVisibleNodes  
 sh::ui::FileView::getSizeFormattingMode  
 sh::ui::FileView::gotoDir (C++ function),  
 sh::ui::FileView::hiddenFilesVisible  
 sh::ui::FileView::historyTracker (C++  
 sh::ui::FileView::iconDimension (C++  
 sh::ui::FileView::index (C++ function), 469,  
 sh::ui::FileView::node (C++ function), 469,  
 sh::ui::FileView::reload (C++ function), 470,  
 sh::ui::FileView::selectedNodes (C++  
 sh::ui::FileView::selectionChanged (C++  
 sh::ui::FileView::setFilemodel (C++ func-  
 sh::ui::FileView::setHiddenFilesVisible  
 sh::ui::FileView::setIconDimension (C++  
 sh::ui::FileView::setSelection (C++ func-  
 sh::ui::FileView::setSizeFormattingMode  
 sh::ui::FileView::setSort (C++ function),  
 sh::ui::FileView::setViewmode (C++ func-  
 sh::ui::FileView::sortColumn (C++ func-  
 sh::ui::FileView::sortOrder (C++ function),  
 sh::ui::FileView::viewmode (C++ function),  
 sh::ui::FileView::viewOptionChanged  
 sh::ui::FileViewMode (C++ enum), 979, 1650  
 sh::ui::FileViewMode::IconView (C++ enu-  
 sh::ui::FileViewMode::ListView (C++ enu-

sh::ui::LogViewDialog (C++ class), 471, 996, 1667  
 sh::ui::LogViewDialog::\_headtext (C++ member), 472, 997, 1668  
 sh::ui::LogViewDialog::\_minseverity (C++ member), 472, 997, 1668  
 sh::ui::LogViewDialog::\_subheadtxt (C++ member), 472, 997, 1668  
 sh::ui::LogViewDialog::close (C++ function), 471, 996, 1668  
 sh::ui::LogViewDialog::dialogId (C++ function), 471, 996, 1667  
 sh::ui::LogViewDialog::headtext (C++ function), 471, 996, 1667  
 sh::ui::LogViewDialog::isInitd (C++ function), 471, 996, 1667  
 sh::ui::LogViewDialog::LogViewDialog (C++ function), 471, 996, 1667  
 sh::ui::LogViewDialog::manager (C++ function), 472, 996, 1668  
 sh::ui::LogViewDialog::minimumSeverity (C++ function), 471, 996, 1667  
 sh::ui::LogViewDialog::subheadtxt (C++ function), 471, 996, 1667  
 sh::ui::LogViewDialog::waitClosed (C++ function), 471, 996, 1668  
 sh::ui::LogViewDialog::wasAccepted (C++ function), 471, 996, 1667  
 sh::ui::LogViewDialog::wasClosed (C++ function), 471, 996, 1668  
 sh::ui::MainWindow (C++ class), 472, 997, 1668  
 sh::ui::MainWindow::\_closeDirectly (C++ function), 475, 1000, 1671  
 sh::ui::MainWindow::\_closing (C++ member), 476, 1000, 1672  
 sh::ui::MainWindow::\_currentProfile (C++ member), 476, 1000, 1672  
 sh::ui::MainWindow::\_detailsPanelVisible (C++ member), 476, 1000, 1672  
 sh::ui::MainWindow::\_errorpanels (C++ member), 476, 1000, 1672  
 sh::ui::MainWindow::\_initialize (C++ function), 472, 997, 1668  
 sh::ui::MainWindow::\_isCloseable (C++ member), 476, 1000, 1672  
 sh::ui::MainWindow::\_mainWindow (C++ member), 476, 1001, 1672  
 sh::ui::MainWindow::\_mainWindow\_runheadless (C++ member), 476, 1001, 1672  
 sh::ui::MainWindow::\_refreshIsCloseable (C++ function), 475, 1000, 1672  
 sh::ui::MainWindow::\_thumbnaillastreqheight (C++ member), 476, 1001, 1672  
 sh::ui::MainWindow::\_thumbnaillastrequestmd (C++ member), 476, 1001, 1672  
 sh::ui::MainWindow::\_thumbnaillastreqwidth (C++ member), 476, 1001, 1672  
 sh::ui::MainWindow::\_thumbnailrequestfollowup (C++ member), 476, 1001, 1672  
 sh::ui::MainWindow::\_thumbnailrequestfollowup\_height (C++ member), 476, 1001, 1672  
 sh::ui::MainWindow::\_thumbnailrequestfollowup\_onAr (C++ member), 476, 1001, 1672  
 sh::ui::MainWindow::\_thumbnailrequestfollowup\_onBe (C++ member), 476, 1001, 1672  
 sh::ui::MainWindow::\_thumbnailrequestfollowup\_widl (C++ member), 476, 1001, 1672  
 sh::ui::MainWindow::\_thumbnailrequestongoing (C++ member), 476, 1001, 1672  
 sh::ui::MainWindow::\_titlePattern (C++ member), 476, 1000, 1672  
 sh::ui::MainWindow::\_treeSticky (C++ member), 476, 1000, 1672  
 sh::ui::MainWindow::\_treeVisible (C++ member), 476, 1000, 1672  
 sh::ui::MainWindow::~MainWindow (C++ function), 472, 997, 1668  
 sh::ui::MainWindow::addFileView (C++ function), 472, 997, 1668  
 sh::ui::MainWindow::addInfoPanel (C++ function), 473, 474, 998, 1670  
 sh::ui::MainWindow::closeApp (C++ function), 475, 1000, 1671  
 sh::ui::MainWindow::createActionExecutionInfoDialog (C++ function), 474, 999, 1671  
 sh::ui::MainWindow::currentDirectory (C++ function), 473, 998, 1669  
 sh::ui::MainWindow::currentFileView (C++ function), 472, 997, 1668  
 sh::ui::MainWindow::currentProfile (C++ function), 473, 998, 1669  
 sh::ui::MainWindow::detailPanelPosition (C++ function), 474, 999, 1670  
 sh::ui::MainWindow::dialogManager (C++ function), 475, 999, 1671  
 sh::ui::MainWindow::fileDetailsPanelVisible (C++ function), 473, 998, 1669  
 sh::ui::MainWindow::fileViewCount (C++ function), 472, 997, 1668  
 sh::ui::MainWindow::fileViewsReloadAll (C++ function), 472, 997, 1668  
 sh::ui::MainWindow::getFileView (C++ function), 472, 997, 1668  
 sh::ui::MainWindow::handleCloseAppRejected (C++ function), 475, 1000, 1671  
 sh::ui::MainWindow::handleClosed (C++ function), 475, 1000, 1671  
 sh::ui::MainWindow::isCloseable (C++

*function*), 475, 1000, 1671

sh::ui::MainWindow::isClosing (C++ *function*), 475, 1000, 1671

sh::ui::MainWindow::isReady (C++ *function*), 475, 1000, 1671

sh::ui::MainWindow::jumpbarIsTextMode (C++ *function*), 474, 999, 1670

sh::ui::MainWindow::jumpbarSetButtonModes (C++ *function*), 474, 999, 1670

sh::ui::MainWindow::jumpbarSetTextMode (C++ *function*), 474, 998, 1670

sh::ui::MainWindow::jumpToEurl (C++ *function*), 473, 998, 1669

sh::ui::MainWindow::jumpToNode (C++ *function*), 473, 998, 1669

sh::ui::MainWindow::KillHelperThread (C++ *class*), 476, 477, 1001, 1672

sh::ui::MainWindow::KillHelperThread::run (C++ *function*), 476, 477, 1001, 1673

sh::ui::MainWindow::killthread (C++ *member*), 476, 1001, 1672

sh::ui::MainWindow::MainWindow (C++ *function*), 472, 997, 1668

sh::ui::MainWindow::mainWindow (C++ *function*), 475, 1000, 1671

sh::ui::MainWindow::onCurrentDirectoryChanged (C++ *function*), 473, 997, 1669

sh::ui::MainWindow::onCurrentFileViewChanged (C++ *function*), 472, 997, 1669

sh::ui::MainWindow::onCurrentProfileChanged (C++ *function*), 473, 998, 1669

sh::ui::MainWindow::onFileViewCountChanged (C++ *function*), 472, 997, 1669

sh::ui::MainWindow::onFileViewOptionsChanged (C++ *function*), 472, 997, 1668

sh::ui::MainWindow::onGlobalViewOptionsChanged (C++ *function*), 473, 997, 1669

sh::ui::MainWindow::reloadProfile (C++ *function*), 473, 998, 1669

sh::ui::MainWindow::removeFileView (C++ *function*), 472, 997, 1668

sh::ui::MainWindow::runsHeadless (C++ *function*), 475, 1000, 1671

sh::ui::MainWindow::setCurrentProfile (C++ *function*), 473, 998, 1669

sh::ui::MainWindow::setDetailPanelPosition (C++ *function*), 474, 999, 1670

sh::ui::MainWindow::setFileDetailsPanelVisible (C++ *function*), 473, 998, 1669

sh::ui::MainWindow::setMainWindow (C++ *function*), 475, 1000, 1671

sh::ui::MainWindow::setTitlePattern (C++ *function*), 473, 998, 1669

sh::ui::MainWindow::setToolBarPosition (C++ *function*), 474, 999, 1670

sh::ui::MainWindow::setTreeSticky (C++ *function*), 473, 998, 1669

sh::ui::MainWindow::setTreeVisible (C++ *function*), 473, 998, 1669

sh::ui::MainWindow::showAboutDialog (C++ *function*), 474, 999, 1670

sh::ui::MainWindow::showErrorPanel (C++ *function*), 475, 999, 1671

sh::ui::MainWindow::showFilePropertyDialog (C++ *function*), 474, 999, 1671

sh::ui::MainWindow::showLogViewDialog (C++ *function*), 474, 999, 1670

sh::ui::MainWindow::showManageBookmarksDialog (C++ *function*), 474, 999, 1671

sh::ui::MainWindow::showManageProfilesDialog (C++ *function*), 474, 999, 1670

sh::ui::MainWindow::showOpenWithDialog (C++ *function*), 474, 999, 1670

sh::ui::MainWindow::showStoreProfileDialog (C++ *function*), 474, 999, 1670

sh::ui::MainWindow::showTuningDialog (C++ *function*), 474, 999, 1670

sh::ui::MainWindow::titlePattern (C++ *function*), 473, 998, 1669

sh::ui::MainWindow::toolbarPosition (C++ *function*), 474, 999, 1670

sh::ui::MainWindow::treeSticky (C++ *function*), 473, 998, 1669

sh::ui::MainWindow::treeVisible (C++ *function*), 473, 998, 1669

sh::ui::MainWindow::uiMode (C++ *function*), 475, 1000, 1671

sh::ui::ManageBookmarksDialog (C++ *class*), 477, 1001, 1673

sh::ui::ManageBookmarksDialog::BM (C++ *struct*), 1886

sh::ui::ManageBookmarksDialog::BM::asBookmark (C++ *function*), 1886

sh::ui::ManageBookmarksDialog::BM::asFolder (C++ *function*), 1886

sh::ui::ManageBookmarksDialog::BM::isFolder (C++ *function*), 1886

sh::ui::ManageBookmarksDialog::BM\_Bookmark (C++ *struct*), 1887

sh::ui::ManageBookmarksDialog::BM\_Bookmark::asBookmark (C++ *function*), 1887

sh::ui::ManageBookmarksDialog::BM\_Bookmark::asFolder (C++ *function*), 1887

sh::ui::ManageBookmarksDialog::BM\_Bookmark::BM\_Bookmark (C++ *function*), 1887

sh::ui::ManageBookmarksDialog::BM\_Bookmark::bookmark (C++ *member*), 1887

sh::ui::ManageBookmarksDialog::BM\_Bookmark::isFolder (C++ *function*), 1886

(C++ function), 1887  
 sh::ui::ManageBookmarksDialog::BM\_Folders (C++ struct), 1887  
 sh::ui::ManageBookmarksDialog::BM\_Folders (C++ function), 1887  
 sh::ui::ManageBookmarksDialog::BM\_Folders (C++ function), 1887  
 sh::ui::ManageBookmarksDialog::BM\_Folders (C++ function), 1887  
 sh::ui::ManageBookmarksDialog::BM\_Folders (C++ member), 1887  
 sh::ui::ManageBookmarksDialog::BM\_Folders (C++ function), 1887  
 sh::ui::ManageBookmarksDialog::BM\_Folders (C++ member), 1887  
 sh::ui::ManageBookmarksDialog::close (C++ function), 477, 1002, 1673  
 sh::ui::ManageBookmarksDialog::dialogId (C++ function), 477, 1001, 1673  
 sh::ui::ManageBookmarksDialog::isInitiated (C++ function), 477, 1001, 1673  
 sh::ui::ManageBookmarksDialog::ManageBookmarksDialog (C++ function), 477, 1001, 1673  
 sh::ui::ManageBookmarksDialog::manager (C++ function), 477, 1002, 1673  
 sh::ui::ManageBookmarksDialog::waitClosed (C++ function), 477, 1002, 1673  
 sh::ui::ManageBookmarksDialog::wasAccepted (C++ function), 477, 1001, 1673  
 sh::ui::ManageBookmarksDialog::wasClosed (C++ function), 477, 1002, 1673  
 sh::ui::ManageProfilesDialog (C++ class), 478, 1002, 1673  
 sh::ui::ManageProfilesDialog::close (C++ function), 478, 1002, 1674  
 sh::ui::ManageProfilesDialog::dialogId (C++ function), 478, 1002, 1674  
 sh::ui::ManageProfilesDialog::isInitiated (C++ function), 478, 1002, 1674  
 sh::ui::ManageProfilesDialog::ManageProfilesDialog (C++ function), 478, 1002, 1674  
 sh::ui::ManageProfilesDialog::manager (C++ function), 478, 1002, 1674  
 sh::ui::ManageProfilesDialog::Node (C++ struct), 1887  
 sh::ui::ManageProfilesDialog::Node::icon (C++ member), 1887  
 sh::ui::ManageProfilesDialog::Node::isGlobal (C++ member), 1887  
 sh::ui::ManageProfilesDialog::Node::nodeId (C++ member), 1887  
 sh::ui::ManageProfilesDialog::Node::pnode (C++ member), 1887  
 sh::ui::ManageProfilesDialog::Node::text (C++ member), 1887  
 sh::ui::ManageProfilesDialog::waitClosed (C++ function), 478, 1002, 1674  
 sh::ui::ManageProfilesDialog::wasAccepted (C++ function), 478, 1002, 1674  
 sh::ui::ManageProfilesDialog::wasClosed (C++ function), 478, 1002, 1674  
 sh::ui::ManageProfilesDialog::withDialog (C++ class), 478, 1003, 1674  
 sh::ui::ManageProfilesDialog::\_node (C++ member), 480, 1004, 1675  
 sh::ui::ManageProfilesDialog::\_openMethods (C++ member), 480, 1004, 1675  
 sh::ui::ManageProfilesDialog::OpenWithDialog::chosenMethod (C++ function), 479, 1003, 1674  
 sh::ui::OpenWithDialog::close (C++ function), 479, 1003, 1675  
 sh::ui::OpenWithDialog::dialogId (C++ function), 479, 1003, 1675  
 sh::ui::OpenWithDialog::isInitiated (C++ function), 479, 1003, 1675  
 sh::ui::OpenWithDialog::manager (C++ function), 479, 1004, 1675  
 sh::ui::OpenWithDialog::node (C++ function), 479, 1003, 1675  
 sh::ui::OpenWithDialog::openMethods (C++ function), 479, 1003, 1675  
 sh::ui::OpenWithDialog::OpenWithDialog (C++ function), 479, 1003, 1674  
 sh::ui::OpenWithDialog::rememberForDirectory (C++ function), 479, 1003, 1674  
 sh::ui::OpenWithDialog::rememberForFile (C++ function), 479, 1003, 1674  
 sh::ui::OpenWithDialog::rememberForMimeType (C++ function), 479, 1003, 1674  
 sh::ui::OpenWithDialog::waitClosed (C++ function), 479, 1003, 1675  
 sh::ui::OpenWithDialog::wasAccepted (C++ function), 479, 1003, 1675  
 sh::ui::OpenWithDialog::wasClosed (C++ function), 479, 1003, 1675  
 sh::ui::qt (C++ type), 1009, 1681, 1777  
 sh::ui::qt::feedbackpanels (C++ type), 1053, 1725, 1822, 1827  
 sh::ui::qt::feedbackpanels::Credentials (C++ class), 537, 1053, 1725, 1822, 1828  
 sh::ui::qt::feedbackpanels::Credentials::~~Credentials (C++ function), 537, 1053, 1725, 1822, 1828  
 sh::ui::qt::feedbackpanels::Credentials::accepted (C++ member), 537, 1054, 1726, 1823, 1828  
 sh::ui::qt::feedbackpanels::Credentials::anonymous (C++ member), 537, 1054, 1726, 1823, 1828  
 sh::ui::qt::feedbackpanels::Credentials::cancelRequest (C++ function), 537, 1053, 1725, 1822, 1828  
 sh::ui::qt::feedbackpanels::Credentials::Credential (C++ function), 537, 1053, 1725, 1822, 1828



sh::ui::qt::feedbackpanels::Credentials:shdomain:qt::feedbackpanels::GridForm::btn2index  
 (C++ member), 537, 1054, 1726, 1823, 1828 (C++ member), 539, 1055, 1727, 1824, 1830  
 sh::ui::qt::feedbackpanels::Credentials:shsClosed:qt::feedbackpanels::GridForm::cancelRequest  
 (C++ function), 537, 1053, 1725, 1822, 1828 (C++ function), 539, 1055, 1727, 1824, 1829  
 sh::ui::qt::feedbackpanels::Credentials:sh:shkAnyFeedbackpanels::GridForm::GridForm  
 (C++ function), 538, 1054, 1726, 1823, 1828 (C++ function), 539, 1055, 1727, 1824, 1829  
 sh::ui::qt::feedbackpanels::Credentials:sh:pushBtnFeedbackpanels::GridForm::isClosed  
 (C++ function), 538, 1054, 1726, 1823, 1828 (C++ function), 539, 1055, 1727, 1824, 1829  
 sh::ui::qt::feedbackpanels::Credentials:sh:pushBtnFeedbackpanels::GridForm::preferredHei  
 (C++ function), 538, 1054, 1726, 1823, 1828 (C++ function), 539, 1055, 1727, 1824, 1829  
 sh::ui::qt::feedbackpanels::Credentials:spassword:qt::feedbackpanels::GridForm::slot\_btncli  
 (C++ member), 537, 1054, 1726, 1823, 1828 (C++ function), 539, 1056, 1728, 1825, 1830  
 sh::ui::qt::feedbackpanels::Credentials:spreferrdHeiFeedbackpanels::GridForm::ui  
 (C++ function), 537, 1053, 1725, 1822, 1828 (C++ member), 539, 1055, 1727, 1824, 1830  
 sh::ui::qt::feedbackpanels::Credentials:shememb:qt::feedbackpanels::GridForm::waitUntilClo  
 (C++ member), 537, 1054, 1726, 1823, 1828 (C++ function), 539, 1055, 1727, 1824, 1829  
 sh::ui::qt::feedbackpanels::Credentials:shi:ui::qt::feedbackpanels::GridForm::wasClosed  
 (C++ member), 538, 1054, 1726, 1823, 1828 (C++ function), 539, 1055, 1727, 1824, 1829  
 sh::ui::qt::feedbackpanels::Credentials:shseniam:qt::feedbackpanels::GridFormInnerSimpleCho  
 (C++ member), 537, 1054, 1726, 1823, 1828 (C++ class), 539, 1056, 1728, 1825, 1830  
 sh::ui::qt::feedbackpanels::Credentials:shwaitUntilClFeedbackpanels::GridFormInnerSimpleCho  
 (C++ function), 537, 1053, 1725, 1822, 1828 (C++ function), 539, 1056, 1728, 1825, 1830  
 sh::ui::qt::feedbackpanels::Credentials:shwasGlos:qt::feedbackpanels::GridFormInnerSimpleCho  
 (C++ function), 537, 1054, 1726, 1823, 1828 (C++ member), 540, 1056, 1728, 1825, 1830  
 sh::ui::qt::feedbackpanels::FeedbackPane\$h:ui::qt::feedbackpanels::GridFormInnerSimpleCho  
 (C++ class), 538, 1054, 1726, 1823, 1828 (C++ member), 540, 1056, 1728, 1825, 1830  
 sh::ui::qt::feedbackpanels::FeedbackPane\$h:~:u:os:qt::feedbackpanels::GridFormInnerSimpleCho  
 (C++ member), 538, 1055, 1727, 1824, 1829 (C++ member), 540, 1056, 1728, 1825, 1830  
 sh::ui::qt::feedbackpanels::FeedbackPane\$h:~:u:os:qt:confidFeedbackpanels::GridFormInnerSimpleCho  
 (C++ member), 538, 1055, 1727, 1824, 1829 (C++ function), 540, 1056, 1728, 1825, 1830  
 sh::ui::qt::feedbackpanels::FeedbackPane\$h:~:mitext:qt::feedbackpanels::GridFormInnerSimpleCho  
 (C++ member), 538, 1055, 1727, 1824, 1829 (C++ function), 539, 1056, 1728, 1825, 1830  
 sh::ui::qt::feedbackpanels::FeedbackPane\$h:cancelRequestFeedbackpanels::GridFormInnerSimpleCho  
 (C++ function), 538, 1054, 1726, 1823, 1829 (C++ member), 540, 1056, 1728, 1825, 1830  
 sh::ui::qt::feedbackpanels::FeedbackPane\$h:FeedbtkPFeedbackpanels::GridFormInnerSimpleCho  
 (C++ function), 538, 1054, 1726, 1823, 1829 (C++ member), 540, 1056, 1728, 1825, 1830  
 sh::ui::qt::feedbackpanels::FeedbackPane\$h:isGloged:qt::feedbackpanels::GridFormInnerSimpleCho  
 (C++ function), 538, 1054, 1726, 1823, 1829 (C++ function), 540, 1056, 1728, 1825, 1830  
 sh::ui::qt::feedbackpanels::FeedbackPane\$h:preferrdHeiFeedbackpanels::GridFormInnerSimpleCho  
 (C++ function), 538, 1054, 1726, 1823, 1829 (C++ function), 540, 1056, 1728, 1825, 1830  
 sh::ui::qt::feedbackpanels::FeedbackPane\$h:waitUqtilFeedbackpanels::GridFormInnerSimpleCho  
 (C++ function), 538, 1054, 1726, 1823, 1829 (C++ member), 540, 1056, 1728, 1825, 1830  
 sh::ui::qt::feedbackpanels::FeedbackPane\$h:wasClqtedfeedbackpanels::GridFormRow  
 (C++ function), 538, 1055, 1727, 1824, 1829 (C++ type), 1053, 1725, 1822, 1828  
 sh::ui::qt::feedbackpanels::GridForm sh::ui::qt::feedbackpanels::MsgBox (C++  
 (C++ class), 538, 1055, 1727, 1824, 1829 class), 540, 1056, 1728, 1825, 1830  
 sh::ui::qt::feedbackpanels::GridForm::~\_cancelBtn:qt::feedbackpanels::MsgBox::~\_cancelBtn  
 (C++ member), 539, 1055, 1727, 1824, 1830 (C++ member), 541, 1057, 1729, 1826, 1831  
 sh::ui::qt::feedbackpanels::GridForm::~\_defaultBtn:qt::feedbackpanels::MsgBox::~\_defaultBtn  
 (C++ member), 539, 1055, 1727, 1824, 1830 (C++ member), 541, 1057, 1729, 1826, 1831  
 sh::ui::qt::feedbackpanels::GridForm::~~GridForm:qt::feedbackpanels::MsgBox::~\_initialFocusW  
 (C++ function), 539, 1055, 1727, 1824, 1829 (C++ member), 541, 1057, 1729, 1826, 1831  
 sh::ui::qt::feedbackpanels::GridForm::answer:ui::qt::feedbackpanels::MsgBox::~\_inputBoxMulti  
 (C++ member), 539, 1055, 1727, 1824, 1829 (C++ member), 541, 1057, 1729, 1826, 1831



sh::ui::qt::QtAboutDialog::on\_btnLicensesClickedqt::QtActionExecutionInfoDialog::setBackgr  
(C++ function), 489, 1010, 1682, 1779 (C++ function), 490, 1011, 1683, 1780

sh::ui::qt::QtAboutDialog::QtAboutDialogsh::ui::qt::QtActionExecutionInfoDialog::setDetails  
(C++ function), 488, 1009, 1681, 1778 (C++ function), 490, 1011, 1682, 1779

sh::ui::qt::QtAboutDialog::ui (C++ mem- sh::ui::qt::QtActionExecutionInfoDialog::setForceFo  
ber), 489, 1010, 1682, 1779 (C++ function), 490, 1011, 1683, 1780

sh::ui::qt::QtAboutDialog::waitClosed sh::ui::qt::QtActionExecutionInfoDialog::setHead  
(C++ function), 489, 1009, 1681, 1778 (C++ function), 490, 1011, 1682, 1779

sh::ui::qt::QtAboutDialog::wasAccepted sh::ui::qt::QtActionExecutionInfoDialog::setLogica  
(C++ function), 489, 1009, 1681, 1778 (C++ function), 490, 1011, 1683, 1780

sh::ui::qt::QtAboutDialog::wasClosed sh::ui::qt::QtActionExecutionInfoDialog::setProgres  
(C++ function), 489, 1010, 1681, 1778 (C++ function), 490, 1011, 1682, 1779

sh::ui::qt::QtActionExecutionInfoDialog sh::ui::qt::QtActionExecutionInfoDialog::simpleCho  
(C++ class), 489, 1010, 1682, 1779 (C++ function), 490, 1011, 1683, 1780

sh::ui::qt::QtActionExecutionInfoDialog:sh::ui::qt::QtActionExecutionInfoDialog::simpleInpu  
(C++ function), 491, 1012, 1683, 1780 (C++ function), 490, 1011, 1683, 1780

sh::ui::qt::QtActionExecutionInfoDialog:sh::ui::qt::QtActionExecutionInfoDialog::simpleMes  
(C++ function), 490, 1011, 1682, 1779 (C++ function), 490, 1011, 1683, 1780

sh::ui::qt::QtActionExecutionInfoDialog:sh::ui::qt::QtActionExecutionInfoDialog::ui  
(C++ function), 490, 1011, 1683, 1780 (C++ member), 491, 1012, 1683, 1780

sh::ui::qt::QtActionExecutionInfoDialog:sh::ui::qt::QtActionExecutionInfoDialog::unixPermis  
(C++ member), 491, 1012, 1683, 1780 (C++ function), 490, 1011, 1683, 1780

sh::ui::qt::QtActionExecutionInfoDialog:sh::ui::qt::QtActionExecutionInfoPanel  
(C++ function), 490, 1011, 1682, 1779 (C++ class), 491, 1012, 1684, 1780

sh::ui::qt::QtActionExecutionInfoDialog:sh::ui::qt::QtActionExecutionInfoPanel::\_check\_inde  
(C++ function), 490, 1011, 1683, 1780 (C++ function), 492, 1013, 1684, 1781

sh::ui::qt::QtActionExecutionInfoDialog:sh::ui::qt::QtActionExecutionInfoPanel::\_indetermin  
(C++ function), 490, 1011, 1683, 1780 (C++ member), 492, 1013, 1684, 1781

sh::ui::qt::QtActionExecutionInfoDialog:sh::ui::qt::QtActionExecutionInfoPanel::\_indetermin  
(C++ function), 490, 1011, 1682, 1779 (C++ member), 492, 1013, 1684, 1781

sh::ui::qt::QtActionExecutionInfoDialog:sh::ui::qt::QtActionExecutionInfoPanel::\_isforegrou  
(C++ enum), 489, 1010, 1682, 1779 (C++ member), 492, 1013, 1685, 1781

sh::ui::qt::QtActionExecutionInfoDialog:sh::ui::qt::QtActionExecutionInfoPanel::\_progressA  
(C++ enumerator), 489, 1010, 1682, 1779 (C++ member), 492, 1013, 1684, 1781

sh::ui::qt::QtActionExecutionInfoDialog:sh::ui::qt::QtActionExecutionInfoPanel::\_progressDe  
(C++ enumerator), 489, 1010, 1682, 1779 (C++ member), 492, 1013, 1684, 1781

sh::ui::qt::QtActionExecutionInfoDialog:sh::ui::qt::QtActionExecutionInfoPanel::\_progressDo  
(C++ enumerator), 490, 1010, 1682, 1779 (C++ member), 492, 1013, 1684, 1781

sh::ui::qt::QtActionExecutionInfoDialog:sh::ui::qt::QtActionExecutionInfoPanel::\_width  
(C++ enumerator), 489, 1010, 1682, 1779 (C++ member), 492, 1013, 1684, 1781

sh::ui::qt::QtActionExecutionInfoDialog:sh::ui::qt::QtActionExecutionInfoPanel::colorBase  
(C++ enumerator), 489, 1010, 1682, 1779 (C++ member), 492, 1013, 1685, 1781

sh::ui::qt::QtActionExecutionInfoDialog:sh::ui::qt::QtActionExecutionInfoPanel::colorFrame  
(C++ enumerator), 489, 1010, 1682, 1779 (C++ member), 492, 1013, 1685, 1782

sh::ui::qt::QtActionExecutionInfoDialog:sh::ui::qt::QtActionExecutionInfoPanel::colorProgre  
(C++ enumerator), 490, 1010, 1682, 1779 (C++ member), 492, 1013, 1685, 1781

sh::ui::qt::QtActionExecutionInfoDialog:sh::ui::qt::QtActionExecutionInfoPanel::colorProgre  
(C++ function), 490, 1011, 1682, 1779 (C++ member), 492, 1013, 1685, 1781

sh::ui::qt::QtActionExecutionInfoDialog:sh::ui::qt::QtActionExecutionInfoPanel::colorText  
(C++ function), 491, 1012, 1684, 1780 (C++ member), 492, 1013, 1685, 1782

sh::ui::qt::QtActionExecutionInfoDialog:sh::ui::qt::QtActionExecutionInfoPanel::colorTextDa  
(C++ function), 491, 1012, 1684, 1780 (C++ member), 492, 1013, 1685, 1782

sh::ui::qt::QtActionExecutionInfoDialog:sh::ui::qt::QtActionExecutionInfoPanel::gradient  
(C++ function), 490, 1011, 1682, 1779 (C++ member), 492, 1013, 1685, 1781



sh::ui::qt::QtActionExecutionInfoPanel::shfoui::qt::QtActionMenu::brandingcolor  
 (C++ member), 492, 1013, 1685, 1781 (C++ member), 493, 1014, 1685, 1782  
 sh::ui::qt::QtActionExecutionInfoPanel::shliui::qt::QtActionMenu::clearAllActions  
 (C++ member), 492, 1013, 1684, 1781 (C++ function), 492, 1013, 1685, 1782  
 sh::ui::qt::QtActionExecutionInfoPanel::shClickedqt::QtActionMenu::QtActionMenu  
 (C++ function), 491, 1012, 1684, 1781 (C++ function), 492, 1013, 1685, 1782  
 sh::ui::qt::QtActionExecutionInfoPanel::shDesiredqt::QtActionMenu::removeActionAt  
 (C++ function), 491, 1012, 1684, 1781 (C++ function), 492, 1013, 1685, 1782  
 sh::ui::qt::QtActionExecutionInfoPanel::shVisibilityQtActionMenu::setHeader  
 (C++ function), 491, 1012, 1684, 1781 (C++ function), 492, 1013, 1685, 1782  
 sh::ui::qt::QtActionExecutionInfoPanel::QtActionExecutionInfoPanelHandler (C++  
 (C++ function), 491, 1012, 1684, 1781 class), 493, 1014, 1686, 1782  
 sh::ui::qt::QtActionExecutionInfoPanel::shForceBreakQtActionMenuHandler::\_append\_actions  
 (C++ function), 491, 1012, 1684, 1781 (C++ function), 493, 1014, 1686, 1783  
 sh::ui::qt::QtActionExecutionInfoPanel::shLabelqt::QtActionMenuHandler::\_applyPropertiesTo  
 (C++ function), 491, 1012, 1684, 1781 (C++ function), 494, 1015, 1686, 1783  
 sh::ui::qt::QtActionExecutionInfoPanel::shPanelVisibilityQtActionMenuHandler::\_createAndConnectA  
 (C++ function), 491, 1012, 1684, 1781 (C++ function), 494, 1015, 1686, 1783  
 sh::ui::qt::QtActionExecutionInfoPanel::shProgressQtActionMenuHandler::\_getDefaultAction  
 (C++ function), 491, 1012, 1684, 1781 (C++ function), 494, 1015, 1686, 1783  
 sh::ui::qt::QtActionExecutionInfoPanel::shWidthqt::QtActionMenuHandler::\_markDefault  
 (C++ function), 491, 1012, 1684, 1781 (C++ function), 494, 1015, 1686, 1783  
 sh::ui::qt::QtActionExecutionInfoPanel::shzeHintqt::QtActionMenuHandler::\_markDefault2  
 (C++ function), 491, 1012, 1684, 1781 (C++ function), 493, 1014, 1686, 1783  
 sh::ui::qt::QtActionMenu (C++ class), 492, sh::ui::qt::QtActionMenuHandler::\_updateSubMenu  
 1013, 1685, 1782 (C++ function), 494, 1015, 1686, 1783  
 sh::ui::qt::QtActionMenu::\_boldfont sh::ui::qt::QtActionMenuHandler::diracts  
 (C++ member), 493, 1014, 1685, 1782 (C++ member), 493, 1014, 1686, 1783  
 sh::ui::qt::QtActionMenu::\_cntInternalActionssh::ui::qt::QtActionMenuHandler::menu  
 (C++ member), 493, 1014, 1685, 1782 (C++ member), 493, 1014, 1686, 1783  
 sh::ui::qt::QtActionMenu::\_correctMenuPositionsh::ui::qt::QtActionMenuHandler::qaction2action  
 (C++ function), 493, 1014, 1685, 1782 (C++ member), 493, 1014, 1686, 1783  
 sh::ui::qt::QtActionMenu::\_header (C++ sh::ui::qt::QtActionMenuHandler::QtActionMenuHandle  
 member), 493, 1014, 1685, 1782 (C++ function), 493, 1014, 1686, 1783  
 sh::ui::qt::QtActionMenu::\_normalfont sh::ui::qt::QtActionMenuHandler::selects  
 (C++ member), 493, 1014, 1685, 1782 (C++ member), 493, 1014, 1686, 1783  
 sh::ui::qt::QtActionMenu::\_observeActionsh::ui::qt::QtDialog (C++ class), 494, 1015,  
 (C++ function), 493, 1014, 1685, 1782 1687, 1783  
 sh::ui::qt::QtActionMenu::\_origMenuPosX sh::ui::qt::QtDialog::QtDialog (C++ func-  
 (C++ member), 493, 1014, 1685, 1782 tion), 494, 1015, 1687, 1784  
 sh::ui::qt::QtActionMenu::\_origMenuPosY sh::ui::qt::QtDialogManager (C++ class),  
 (C++ member), 493, 1014, 1685, 1782 494, 1015, 1687, 1784  
 sh::ui::qt::QtActionMenu::actionIsHeadersh::ui::qt::QtDialogManager::createAndShowDialog  
 (C++ function), 492, 1013, 1685, 1782 (C++ function), 495, 1015, 1687, 1784  
 sh::ui::qt::QtActionMenu::addNewAction sh::ui::qt::QtDialogManager::getAllDialogIds  
 (C++ function), 492, 1013, 1685, 1782 (C++ function), 494, 1015, 1687, 1784  
 sh::ui::qt::QtActionMenu::addNewHeader sh::ui::qt::QtDialogManager::getAllDialogs  
 (C++ function), 492, 1013, 1685, 1782 (C++ function), 494, 1015, 1687, 1784  
 sh::ui::qt::QtActionMenu::addNewSeparatorsh::ui::qt::QtDialogManager::getDialogById  
 (C++ function), 492, 1013, 1685, 1782 (C++ function), 494, 1015, 1687, 1784  
 sh::ui::qt::QtActionMenu::addNewSubMenu sh::ui::qt::QtDialogManager::showDialog  
 (C++ function), 492, 1013, 1685, 1782 (C++ function), 495, 1016, 1688, 1784  
 sh::ui::qt::QtActionMenu::allActions sh::ui::qt::QtExceptionDialog (C++ class),  
 (C++ function), 492, 1013, 1685, 1782 495, 1016, 1688, 1784

sh::ui::qt::QtExceptionDialog::_answer (C++ member), 496, 1017, 1689, 1786	sh::ui::qt::QtFileDetailsPanel (C++ class), 497, 1017, 1689, 1786
sh::ui::qt::QtExceptionDialog::~QtExceptionDialog (C++ function), 495, 1016, 1688, 1785	sh::ui::qt::QtFileDetailsPanel::_nodes (C++ member), 497, 1018, 1690, 1786
sh::ui::qt::QtExceptionDialog::answer (C++ function), 495, 1016, 1688, 1785	sh::ui::qt::QtFileDetailsPanel::_orientation (C++ member), 497, 1018, 1690, 1786
sh::ui::qt::QtExceptionDialog::close (C++ function), 496, 1017, 1689, 1785	sh::ui::qt::QtFileDetailsPanel::_panelDetails (C++ member), 497, 1018, 1690, 1786
sh::ui::qt::QtExceptionDialog::details (C++ function), 495, 1016, 1688, 1785	sh::ui::qt::QtFileDetailsPanel::_placements (C++ member), 497, 1018, 1690, 1786
sh::ui::qt::QtExceptionDialog::dialogId (C++ function), 496, 1016, 1688, 1785	sh::ui::qt::QtFileDetailsPanel::_widgetimagecache (C++ member), 497, 1018, 1690, 1786
sh::ui::qt::QtExceptionDialog::error1 (C++ function), 495, 1016, 1688, 1785	sh::ui::qt::QtFileDetailsPanel::_widgetimagecachet (C++ member), 497, 1018, 1690, 1786
sh::ui::qt::QtExceptionDialog::error2 (C++ function), 495, 1016, 1688, 1785	sh::ui::qt::QtFileDetailsPanel::~QtFileDetailsPanel (C++ function), 497, 1018, 1689, 1786
sh::ui::qt::QtExceptionDialog::icon (C++ function), 495, 1016, 1688, 1785	sh::ui::qt::QtFileDetailsPanel::DetailPlacement (C++ class), 497, 498, 1018, 1690, 1787
sh::ui::qt::QtExceptionDialog::isInitd (C++ function), 496, 1017, 1688, 1785	sh::ui::qt::QtFileDetailsPanel::DetailPlacement::de (C++ member), 497, 498, 1018, 1690, 1787
sh::ui::qt::QtExceptionDialog::manager (C++ function), 496, 1017, 1689, 1785	sh::ui::qt::QtFileDetailsPanel::DetailPlacement::De (C++ function), 497, 498, 1018, 1690, 1787
sh::ui::qt::QtExceptionDialog::mayCancel (C++ function), 495, 1016, 1688, 1785	sh::ui::qt::QtFileDetailsPanel::DetailPlacement::h (C++ member), 497, 498, 1018, 1690, 1787
sh::ui::qt::QtExceptionDialog::mayClose (C++ function), 495, 1016, 1688, 1785	sh::ui::qt::QtFileDetailsPanel::DetailPlacement::ro (C++ member), 497, 498, 1018, 1690, 1787
sh::ui::qt::QtExceptionDialog::mayRetry (C++ function), 495, 1016, 1688, 1785	sh::ui::qt::QtFileDetailsPanel::DetailPlacement::w (C++ member), 497, 498, 1018, 1690, 1787
sh::ui::qt::QtExceptionDialog::on_btnCancel (C++ function), 496, 1017, 1689, 1786	sh::ui::qt::QtFileDetailsPanel::DetailPlacement::x (C++ member), 497, 498, 1018, 1690, 1787
sh::ui::qt::QtExceptionDialog::on_btnClose (C++ function), 496, 1017, 1689, 1786	sh::ui::qt::QtFileDetailsPanel::DetailPlacement::y (C++ member), 497, 498, 1018, 1690, 1787
sh::ui::qt::QtExceptionDialog::on_btnExist (C++ function), 496, 1017, 1689, 1786	sh::ui::qt::QtFileDetailsPanel::DetailRowPlacement (C++ class), 497, 498, 1018, 1690, 1787
sh::ui::qt::QtExceptionDialog::on_btnExistAndSave (C++ function), 496, 1017, 1689, 1786	sh::ui::qt::QtFileDetailsPanel::DetailRowPlacement (C++ member), 498, 499, 1019, 1691, 1787
sh::ui::qt::QtExceptionDialog::on_btnRetry (C++ function), 496, 1017, 1689, 1786	sh::ui::qt::QtFileDetailsPanel::DetailRowPlacement (C++ member), 498, 499, 1019, 1691, 1787
sh::ui::qt::QtExceptionDialog::on_btnShowDetails (C++ function), 496, 1017, 1689, 1786	sh::ui::qt::QtFileDetailsPanel::DetailRowPlacement (C++ function), 498, 499, 1019, 1690, 1787
sh::ui::qt::QtExceptionDialog::on_btnShowLog (C++ function), 496, 1017, 1689, 1786	sh::ui::qt::QtFileDetailsPanel::DetailRowPlacement (C++ member), 498, 499, 1019, 1691, 1787
sh::ui::qt::QtExceptionDialog::QtExceptionDialog (C++ function), 495, 1016, 1688, 1785	sh::ui::qt::QtFileDetailsPanel::DetailRowPlacement (C++ member), 498, 499, 1019, 1691, 1787
sh::ui::qt::QtExceptionDialog::showLogLabelAndDetail (C++ function), 496, 1016, 1688, 1785	sh::ui::qt::QtFileDetailsPanel::fontBold (C++ member), 497, 1018, 1690, 1786
sh::ui::qt::QtExceptionDialog::ui (C++ member), 496, 1017, 1689, 1786	sh::ui::qt::QtFileDetailsPanel::fontNormal (C++ member), 497, 1018, 1690, 1786
sh::ui::qt::QtExceptionDialog::waitClosed (C++ function), 496, 1017, 1688, 1785	sh::ui::qt::QtFileDetailsPanel::PADDINGX (C++ member), 497, 1018, 1690, 1786
sh::ui::qt::QtExceptionDialog::wasAccepted (C++ function), 496, 1017, 1688, 1785	sh::ui::qt::QtFileDetailsPanel::PADDINGY (C++ member), 497, 1018, 1690, 1786
sh::ui::qt::QtExceptionDialog::wasClosed (C++ function), 496, 1017, 1689, 1785	sh::ui::qt::QtFileDetailsPanel::QtFileDetailsPanel (C++ function), 497, 1018, 1689, 1786

sh::ui::qt::QtFileDetailsPanel::setNodeSSH::ui::qt::QtFileIconview::setBackgroundColor  
 (C++ function), 497, 1018, 1689, 1786 (C++ function), 499, 1019, 1691, 1788  
 sh::ui::qt::QtFileDetailsPanel::setOrientationsh::ui::qt::QtFileIconview::setHiddenFilesVisible  
 (C++ function), 497, 1018, 1689, 1786 (C++ function), 499, 1019, 1691, 1788  
 sh::ui::qt::QtFileDetailsPanel::sizeHintsh::ui::qt::QtFileIconview::setSelection  
 (C++ function), 497, 1018, 1689, 1786 (C++ function), 499, 1019, 1691, 1788  
 sh::ui::qt::QtFileIconview (C++ class), 499, sh::ui::qt::QtFileIconview::setSizeFormatting  
 1019, 1691, 1787 (C++ function), 499, 1019, 1691, 1788  
 sh::ui::qt::QtFileIconview::\_deleg\_h sh::ui::qt::QtFileIconview::setSort  
 (C++ member), 500, 1020, 1692, 1788 (C++ function), 499, 1019, 1691, 1788  
 sh::ui::qt::QtFileIconview::\_deleg\_w sh::ui::qt::QtFileIconview::setThumbDimension  
 (C++ member), 500, 1020, 1692, 1788 (C++ function), 499, 1019, 1691, 1788  
 sh::ui::qt::QtFileIconview::\_dsx (C++ sh::ui::qt::QtFileIconview::thread (C++  
 member), 500, 1020, 1692, 1788 function), 499, 1019, 1691, 1788  
 sh::ui::qt::QtFileIconview::\_dsy (C++ sh::ui::qt::QtFileList (C++ class), 500, 1020,  
 member), 500, 1020, 1692, 1788 1692, 1788  
 sh::ui::qt::QtFileIconview::\_lineheight sh::ui::qt::QtFileList::\_adaptColumnWidth  
 (C++ member), 500, 1020, 1692, 1788 (C++ function), 501, 1021, 1693, 1789  
 sh::ui::qt::QtFileIconview::as\_qabstractshemviewqt::QtFileList::\_skip\_slot\_sectionResized  
 (C++ function), 499, 1019, 1691, 1788 (C++ member), 501, 1021, 1693, 1789  
 sh::ui::qt::QtFileIconview::as\_qobject sh::ui::qt::QtFileList::as\_qabstractitemview  
 (C++ function), 499, 1019, 1691, 1788 (C++ function), 501, 1020, 1692, 1789  
 sh::ui::qt::QtFileIconview::as\_qwidget sh::ui::qt::QtFileList::as\_qobject (C++  
 (C++ function), 499, 1019, 1691, 1788 function), 501, 1020, 1692, 1789  
 sh::ui::qt::QtFileIconview::backgroundColsh::ui::qt::QtFileList::as\_qwidget (C++  
 (C++ function), 499, 1019, 1691, 1788 function), 501, 1020, 1692, 1789  
 sh::ui::qt::QtFileIconview::configure sh::ui::qt::QtFileList::backgroundColor  
 (C++ function), 499, 1019, 1691, 1788 (C++ function), 500, 1020, 1692, 1789  
 sh::ui::qt::QtFileIconview::control sh::ui::qt::QtFileList::cfgvalFileListIconSize  
 (C++ function), 499, 1019, 1691, 1788 (C++ member), 501, 1021, 1693, 1790  
 sh::ui::qt::QtFileIconview::focus (C++ sh::ui::qt::QtFileList::configure (C++  
 function), 499, 1019, 1691, 1788 function), 501, 1020, 1692, 1789  
 sh::ui::qt::QtFileIconview::getAllVisibleNodessh::ui::qt::QtFileList::control (C++  
 (C++ function), 499, 1019, 1691, 1788 function), 501, 1020, 1692, 1789  
 sh::ui::qt::QtFileIconview::gotoDir sh::ui::qt::QtFileList::focus (C++ func-  
 (C++ function), 499, 1019, 1691, 1788 tion), 501, 1020, 1692, 1789  
 sh::ui::qt::QtFileIconview::MyItemDelegash::ui::qt::QtFileList::getAllVisibleNodes  
 (C++ class), 500 (C++ function), 501, 1020, 1692, 1789  
 sh::ui::qt::QtFileIconview::MyItemDelegash::ui::qt::QtFileList::getVisibleFileCountgetColumnNameForIndex  
 (C++ function), 500 (C++ function), 501, 1021, 1693, 1789  
 sh::ui::qt::QtFileIconview::MyItemDelegash::ui::qt::QtFileList::gotoDir (C++  
 (C++ member), 500 function), 501, 1020, 1692, 1789  
 sh::ui::qt::QtFileIconview::MyItemDelegash::ui::qt::QtFileList::node (C++ func-  
 (C++ function), 500 tion), 500, 1020, 1692, 1789  
 sh::ui::qt::QtFileIconview::MyItemDelegash::paintqt::QtFileList::QtFileList (C++  
 (C++ function), 500 function), 500, 1020, 1692, 1789  
 sh::ui::qt::QtFileIconview::MyItemDelegash::sizeHintQtFileList::selectedNodes  
 (C++ function), 500 (C++ function), 500, 1020, 1692, 1789  
 sh::ui::qt::QtFileIconview::node (C++ sh::ui::qt::QtFileList::setBackgroundColor  
 function), 499, 1019, 1691, 1788 (C++ function), 500, 1020, 1692, 1789  
 sh::ui::qt::QtFileIconview::QtFileIconviewsh::ui::qt::QtFileList::setHiddenFilesVisible  
 (C++ function), 499, 1019, 1691, 1788 (C++ function), 501, 1020, 1692, 1789  
 sh::ui::qt::QtFileIconview::selectedNodeSSH::ui::qt::QtFileList::setSelection  
 (C++ function), 499, 1019, 1691, 1788 (C++ function), 500, 1020, 1692, 1789

```

sh::ui::qt::QtFileList::setSizeFormattingh::ui::qt::QtFilePropertyDialog::ui
    (C++ function), 501, 1020, 1692, 1789          (C++ member), 503, 1023, 1695, 1791
sh::ui::qt::QtFileList::setSort      (C++ sh::ui::qt::QtFilePropertyDialog::waitClosed
    function), 500, 1020, 1692, 1789          (C++ function), 502, 1022, 1694, 1790
sh::ui::qt::QtFileList::slot_sectionResizedsh::ui::qt::QtFilePropertyDialog::wasAccepted
    (C++ function), 501, 1021, 1693, 1789          (C++ function), 502, 1022, 1694, 1790
sh::ui::qt::QtFileList::thread (C++ func- sh::ui::qt::QtFilePropertyDialog::wasClosed
    tion), 501, 1020, 1692, 1789          (C++ function), 502, 1022, 1694, 1791
sh::ui::qt::QtFilePropertyDialog      (C++ sh::ui::qt::QtFilePropertyDialog::widgetAt
    class), 501, 1021, 1693, 1790          (C++ function), 502, 1021, 1693, 1790
sh::ui::qt::QtFilePropertyDialog::_internsh::ui::qt::QtFilePropertyDialog::widgetCount
    (C++ member), 503, 1023, 1695, 1791          (C++ function), 502, 1021, 1693, 1790
sh::ui::qt::QtFilePropertyDialog::_skip_sh::ui::qt::QtFilePropertyDialog::widgetCount
    (C++ member), 503, 1023, 1695, 1791          (C++ class), 503, 1023, 1695, 1792
sh::ui::qt::QtFilePropertyDialog::_skip_sh::ui::qt::QtFilePropertyDialog::widgetCount
    (C++ member), 503, 1023, 1695, 1791          (C++ function), 504, 1023, 1695, 1792
sh::ui::qt::QtFilePropertyDialog::_tabheadsh::ui::qt::QtFilePropertyDialogTabActionsView::cor
    (C++ member), 503, 1023, 1695, 1791          (C++ function), 504, 1023, 1695, 1792
sh::ui::qt::QtFilePropertyDialog::~QtFilesh::ui::qt::QtFilePropertyDialogTabActionsView::onF
    (C++ function), 502, 1021, 1693, 1790          (C++ function), 504, 1023, 1695, 1792
sh::ui::qt::QtFilePropertyDialog::addTabsh::ui::qt::QtFilePropertyDialogTabActionsView::QtF
    (C++ function), 503, 1022, 1694, 1791          (C++ function), 504, 1023, 1695, 1792
sh::ui::qt::QtFilePropertyDialog::close sh::ui::qt::QtFilePropertyDialogTabActionsView::set
    (C++ function), 502, 1022, 1694, 1790          (C++ function), 504, 1023, 1695, 1792
sh::ui::qt::QtFilePropertyDialog::createSh::ui::qt::QtFilePropertyDialogTabActionsView::set
    (C++ function), 502, 1021, 1693, 1790          (C++ function), 504, 1023, 1695, 1792
sh::ui::qt::QtFilePropertyDialog::createSh::ui::qt::QtFilePropertyDialogTabActionsView::set
    (C++ function), 502, 1021, 1693, 1790          (C++ function), 504, 1023, 1695, 1792
sh::ui::qt::QtFilePropertyDialog::createSh::ui::qt::QtFilePropertyDialogTabIconTextBannerV
    (C++ function), 502, 1021, 1693, 1790          (C++ class), 504, 1023, 1695, 1792
sh::ui::qt::QtFilePropertyDialog::dialogSh::ui::qt::QtFilePropertyDialogTabIconTextBannerV
    (C++ function), 502, 1021, 1693, 1790          (C++ member), 504, 1024, 1696, 1793
sh::ui::qt::QtFilePropertyDialog::init sh::ui::qt::QtFilePropertyDialogTabIconTextBannerV
    (C++ function), 502, 1021, 1693, 1790          (C++ function), 504, 1024, 1696, 1792
sh::ui::qt::QtFilePropertyDialog::isInitsh::ui::qt::QtFilePropertyDialogTabIconTextBannerV
    (C++ function), 502, 1022, 1694, 1790          (C++ function), 504, 1024, 1696, 1792
sh::ui::qt::QtFilePropertyDialog::managesh::ui::qt::QtFilePropertyDialogTabIconTextBannerV
    (C++ function), 502, 1022, 1694, 1791          (C++ function), 504, 1024, 1696, 1792
sh::ui::qt::QtFilePropertyDialog::on_btnSh::ui::qt::QtFilePropertyDialogTabIconTextBannerV
    (C++ function), 503, 1022, 1694, 1791          (C++ function), 504, 1024, 1696, 1792
sh::ui::qt::QtFilePropertyDialog::on_btnRefreshsh::ui::qt::QtFilePropertyDialogTabTableView
    (C++ function), 503, 1022, 1694, 1791          (C++ class), 505, 1024, 1696, 1793
sh::ui::qt::QtFilePropertyDialog::on_tabSh::ui::qt::QtFilePropertyDialogTabTableView::creat
    (C++ function), 503, 1022, 1694, 1791          (C++ function), 505, 1024, 1696, 1793
sh::ui::qt::QtFilePropertyDialog::QtFilesh::ui::qt::QtFilePropertyDialogTabTableView::getIt
    (C++ function), 502, 1021, 1693, 1790          (C++ function), 505, 1024, 1696, 1793
sh::ui::qt::QtFilePropertyDialog::refreshsh::ui::qt::QtFilePropertyDialogTabTableView::getSe
    (C++ function), 502, 1021, 1693, 1790          (C++ function), 505, 1024, 1696, 1793
sh::ui::qt::QtFilePropertyDialog::scrollAnealayoqt::QtFilePropertyDialogTabTableView::Item
    (C++ member), 503, 1023, 1695, 1791          (C++ type), 505, 1024, 1696, 1793
sh::ui::qt::QtFilePropertyDialog::selectTabByScrqt::QtFilePropertyDialogTabTableView::QtFi
    (C++ function), 503, 1022, 1694, 1791          (C++ function), 505, 1024, 1696, 1793
sh::ui::qt::QtFilePropertyDialog::tabs sh::ui::qt::QtFilePropertyDialogTabTableView::setCo
    (C++ function), 502, 1021, 1693, 1790          (C++ function), 505, 1024, 1696, 1793

```



sh::ui::qt::QtFilePropertyDialogTabTextVshw:ui::qt::QtFilesystemPanel::selectFolder  
 (C++ class), 505, 1024, 1696, 1793 (C++ function), 510, 1025, 1697, 1794  
 sh::ui::qt::QtFilePropertyDialogTabTextVshw:ui::qt::QtFilePropertyDialogTabTextVCustomWidget  
 (C++ function), 505, 1024, 1696, 1793 (C++ function), 510, 1025, 1697, 1794  
 sh::ui::qt::QtFilePropertyDialogTabTextVshw:ui::qt::QtFilesystemPanel::ui (C++  
 (C++ function), 505, 1024, 1696, 1793 member), 510, 1025, 1697, 1794  
 sh::ui::qt::QtFilesystemPanel (C++ class), sh::ui::qt::QtFilesystemPanel::view  
 510, 1024, 1696, 1793 (C++ function), 510, 1025, 1697, 1793  
 sh::ui::qt::QtFilesystemPanel::\_customwidgetsh:ui::qt::QtFilesystemPanel::viewsCount  
 (C++ member), 510, 1025, 1697, 1794 (C++ function), 510, 1025, 1697, 1793  
 sh::ui::qt::QtFilesystemPanel::\_emit\_viewsh:ui::qt::QtFilesystemPanel::viewwidget  
 (C++ function), 510, 1025, 1697, 1794 (C++ function), 510, 1025, 1697, 1794  
 sh::ui::qt::QtFilesystemPanel::\_mainviewsh:ui::qt::QtFileView (C++ class), 505, 1026,  
 (C++ member), 510, 1025, 1697, 1794 1698, 1794  
 sh::ui::qt::QtFilesystemPanel::\_skip\_evensh:ui::qt::QtFileView::\_async\_gotodir\_index  
 (C++ member), 511, 1025, 1697, 1794 (C++ member), 506, 1026, 1698, 1795  
 sh::ui::qt::QtFilesystemPanel::\_viewcontsh:ui::qt::QtFileView::\_dragIsValid  
 (C++ member), 511, 1025, 1697, 1794 (C++ member), 506, 1026, 1698, 1795  
 sh::ui::qt::QtFilesystemPanel::\_views sh:ui::qt::QtFileView::\_dragStartPosition  
 (C++ member), 510, 1025, 1697, 1794 (C++ member), 506, 1026, 1698, 1795  
 sh::ui::qt::QtFilesystemPanel::~QtFilesystemPanelsh:ui::qt::QtFileView::\_sort\_column  
 (C++ function), 510, 1025, 1697, 1793 (C++ member), 506, 1026, 1698, 1795  
 sh::ui::qt::QtFilesystemPanel::activeColsh:ui::qt::QtFileView::\_sort\_order  
 (C++ member), 510, 1025, 1697, 1794 (C++ member), 506, 1026, 1698, 1795  
 sh::ui::qt::QtFilesystemPanel::activeViewsh:ui::qt::QtFileView::\_tmp\_bgColor  
 (C++ member), 510, 1025, 1697, 1794 (C++ member), 506, 1026, 1698, 1795  
 sh::ui::qt::QtFilesystemPanel::addView sh:ui::qt::QtFileView::\_viewctrl (C++  
 (C++ function), 510, 1025, 1697, 1793 member), 506, 1026, 1698, 1795  
 sh::ui::qt::QtFilesystemPanel::buildViewsh:ui::qt::QtFileView::~QtFileView  
 (C++ function), 510, 1025, 1697, 1794 (C++ function), 506, 1026, 1698, 1795  
 sh::ui::qt::QtFilesystemPanel::currentViewsh:ui::qt::QtFileView::as\_qabstractitemview  
 (C++ function), 510, 1025, 1697, 1793 (C++ function), 506, 1026, 1698, 1795  
 sh::ui::qt::QtFilesystemPanel::currentViewIndexsh:ui::qt::QtFileView::as\_qobject (C++  
 (C++ function), 510, 1025, 1697, 1793 function), 506, 1026, 1698, 1795  
 sh::ui::qt::QtFilesystemPanel::inactiveColsh:ui::qt::QtFileView::as\_qwidget (C++  
 (C++ member), 510, 1025, 1697, 1794 function), 506, 1026, 1698, 1795  
 sh::ui::qt::QtFilesystemPanel::panelCurrentViewsh:ui::qt::QtFileView::backgroundColor  
 (C++ function), 510, 1025, 1697, 1794 (C++ function), 506, 1026, 1698, 1795  
 sh::ui::qt::QtFilesystemPanel::panelFolderActivatedsh:ui::qt::QtFileView::configure (C++  
 (C++ function), 510, 1025, 1697, 1794 function), 506, 1026, 1698, 1795  
 sh::ui::qt::QtFilesystemPanel::panelSelectionsh:ui::qt::QtFileView::control (C++  
 (C++ function), 510, 1025, 1697, 1794 function), 506, 1026, 1698, 1795  
 sh::ui::qt::QtFilesystemPanel::panelViewsh:ui::qt::QtFileView::EventFilter  
 (C++ function), 510, 1025, 1697, 1794 (C++ class), 507, 509, 1027, 1699, 1796  
 sh::ui::qt::QtFilesystemPanel::panelViewsh:ui::qt::QtFileView::EventFilter::EventFilter  
 (C++ function), 510, 1025, 1697, 1794 (C++ function), 507, 509, 1027, 1699, 1796  
 sh::ui::qt::QtFilesystemPanel::panelViewOptionsh:ui::qt::QtFileView::EventFilter::eventFilter  
 (C++ function), 510, 1025, 1697, 1794 (C++ function), 507, 509, 1027, 1699, 1796  
 sh::ui::qt::QtFilesystemPanel::QtFilesystemPanelsh:ui::qt::QtFileView::EventFilter::fileview  
 (C++ function), 510, 1025, 1697, 1793 (C++ member), 507, 509, 1027, 1699, 1796  
 sh::ui::qt::QtFilesystemPanel::removeViewsh:ui::qt::QtFileView::EventFilter::fileviewctrl  
 (C++ function), 510, 1025, 1697, 1793 (C++ member), 507, 509, 1027, 1699, 1796  
 sh::ui::qt::QtFilesystemPanel::selectedNsh:ui::qt::QtFileView::focus (C++ func-  
 (C++ function), 510, 1025, 1697, 1794 tion), 506, 1026, 1698, 1795

```

sh::ui::qt::QtFileView::getAllVisibleNodes (C++ function), 506, 1026, 1698, 1795
sh::ui::qt::QtFileView::gotoDir (C++ function), 506, 1026, 1698, 1795
sh::ui::qt::QtFileView::node (C++ function), 506, 1026, 1698, 1795
sh::ui::qt::QtFileView::QtFileView (C++ function), 506, 1026, 1698, 1795
sh::ui::qt::QtFileView::selectedNodes (C++ function), 506, 1026, 1698, 1795
sh::ui::qt::QtFileView::setBackgroundColors (C++ function), 506, 1026, 1698, 1795
sh::ui::qt::QtFileView::setHiddenFilesVisible (C++ function), 506, 1026, 1698, 1795
sh::ui::qt::QtFileView::setSelection (C++ function), 506, 1026, 1698, 1795
sh::ui::qt::QtFileView::setSizeFormattingMode (C++ function), 506, 1026, 1698, 1795
sh::ui::qt::QtFileView::setSort (C++ function), 506, 1026, 1698, 1795
sh::ui::qt::QtFileView::thread (C++ function), 506, 1026, 1698, 1795
sh::ui::qt::QtFileViewControl (C++ class), 507, 1027, 1699, 1796
sh::ui::qt::QtFileViewControl::_columnDimensions (C++ member), 509, 1029, 1701, 1798
sh::ui::qt::QtFileViewControl::_hiddenFilesVisible (C++ member), 509, 1029, 1701, 1798
sh::ui::qt::QtFileViewControl::_historyTracker (C++ member), 509, 1029, 1701, 1798
sh::ui::qt::QtFileViewControl::_iconDimensions (C++ member), 509, 1029, 1701, 1798
sh::ui::qt::QtFileViewControl::_panel (C++ member), 509, 1029, 1701, 1798
sh::ui::qt::QtFileViewControl::_sizeFormattingMode (C++ member), 509, 1029, 1701, 1798
sh::ui::qt::QtFileViewControl::_sortColumn (C++ member), 509, 1029, 1701, 1798
sh::ui::qt::QtFileViewControl::_sortOrder (C++ member), 509, 1029, 1701, 1798
sh::ui::qt::QtFileViewControl::_viewModes (C++ member), 509, 1029, 1701, 1798
sh::ui::qt::QtFileViewControl::~QtFileViewControl (C++ function), 507, 1027, 1699, 1796
sh::ui::qt::QtFileViewControl::columnDimensions (C++ function), 507, 1027, 1699, 1796
sh::ui::qt::QtFileViewControl::emit_nodesActivated (C++ function), 508, 1028, 1700, 1796
sh::ui::qt::QtFileViewControl::emit_selectionChanged (C++ function), 508, 1028, 1700, 1797
sh::ui::qt::QtFileViewControl::fileModel (C++ function), 508, 1028, 1700, 1797
sh::ui::qt::QtFileViewControl::getAllVisibleNodes (C++ function), 508, 1028, 1700, 1797
sh::ui::qt::QtFileViewControl::getSizeFormattingMode (C++ function), 507, 1027, 1699, 1796
sh::ui::qt::QtFileViewControl::gotoDir (C++ function), 507, 1027, 1699, 1796
sh::ui::qt::QtFileViewControl::hiddenFilesVisible (C++ function), 508, 1028, 1700, 1796
sh::ui::qt::QtFileViewControl::historyTracker (C++ function), 507, 1027, 1699, 1796
sh::ui::qt::QtFileViewControl::i (C++ member), 509, 1029, 1701, 1798
sh::ui::qt::QtFileViewControl::iconDimension (C++ function), 508, 1028, 1700, 1797
sh::ui::qt::QtFileViewControl::index (C++ function), 507, 1027, 1699, 1796
sh::ui::qt::QtFileViewControl::node (C++ function), 508, 1028, 1700, 1797
sh::ui::qt::QtFileViewControl::nodesActivated (C++ function), 508, 1028, 1700, 1797
sh::ui::qt::QtFileViewControl::QtFileViewControl (C++ function), 507, 1027, 1699, 1796
sh::ui::qt::QtFileViewControl::reload (C++ function), 508, 1028, 1700, 1797
sh::ui::qt::QtFileViewControl::selectedNodes (C++ function), 508, 1028, 1700, 1797
sh::ui::qt::QtFileViewControl::selectionChanged (C++ function), 508, 1028, 1700, 1797
sh::ui::qt::QtFileViewControl::setFileModel (C++ function), 508, 1028, 1700, 1797
sh::ui::qt::QtFileViewControl::setHiddenFilesVisible (C++ function), 508, 1028, 1700, 1797
sh::ui::qt::QtFileViewControl::setIconDimension (C++ function), 508, 1028, 1700, 1797
sh::ui::qt::QtFileViewControl::setIndex (C++ function), 509, 1029, 1701, 1798
sh::ui::qt::QtFileViewControl::setSelection (C++ function), 508, 1028, 1700, 1797
sh::ui::qt::QtFileViewControl::setSizeFormattingMode (C++ function), 507, 1027, 1699, 1796
sh::ui::qt::QtFileViewControl::setSort (C++ function), 508, 1028, 1700, 1797
sh::ui::qt::QtFileViewControl::setViewMode (C++ function), 507, 1027, 1699, 1796
sh::ui::qt::QtFileViewControl::sortColumn (C++ function), 508, 1028, 1700, 1797
sh::ui::qt::QtFileViewControl::sortOrder (C++ function), 508, 1028, 1700, 1797
sh::ui::qt::QtFileViewControl::viewMode (C++ function), 507, 1027, 1699, 1796
sh::ui::qt::QtFileViewControl::viewOptionChanged (C++ function), 508, 1028, 1700, 1797
sh::ui::qt::QtJumpBar (C++ class), 511, 1029, 1701, 1798
sh::ui::qt::QtJumpBar::_buildButtons (C++ function), 511, 1030, 1702, 1799

```

sh::ui::qt::QtJumpBar::\_isTextMode (C++ member), 511, 1030, 1702, 1799  
 sh::ui::qt::QtJumpBar::\_node (C++ member), 511, 1030, 1702, 1799  
 sh::ui::qt::QtJumpBar::~QtJumpBar (C++ function), 511, 1029, 1701, 1798  
 sh::ui::qt::QtJumpBar::eurlRequested (C++ function), 511, 1030, 1702, 1799  
 sh::ui::qt::QtJumpBar::isTextMode (C++ function), 511, 1029, 1701, 1798  
 sh::ui::qt::QtJumpBar::node (C++ function), 511, 1029, 1701, 1798  
 sh::ui::qt::QtJumpBar::nodeChanged (C++ function), 511, 1030, 1702, 1799  
 sh::ui::qt::QtJumpBar::on\_btnAbort\_clicked (C++ function), 512, 1030, 1702, 1799  
 sh::ui::qt::QtJumpBar::on\_btnOk\_clicked (C++ function), 512, 1030, 1702, 1799  
 sh::ui::qt::QtJumpBar::QtJumpBar (C++ function), 511, 1029, 1701, 1798  
 sh::ui::qt::QtJumpBar::setButtonMode (C++ function), 511, 1029, 1701, 1798  
 sh::ui::qt::QtJumpBar::setNode (C++ function), 511, 1029, 1701, 1798  
 sh::ui::qt::QtJumpBar::setTextMode (C++ function), 511, 1029, 1701, 1798  
 sh::ui::qt::QtJumpBar::sizeHint (C++ function), 511, 1029, 1701, 1798  
 sh::ui::qt::QtJumpBar::ui (C++ member), 511, 1030, 1702, 1799  
 sh::ui::qt::QtLinkButton (C++ class), 512, 1030, 1702, 1799  
 sh::ui::qt::QtLinkButton::\_linkcolor (C++ member), 512, 1030, 1702, 1799  
 sh::ui::qt::QtLinkButton::\_menuchangetxt (C++ member), 512, 1030, 1702, 1799  
 sh::ui::qt::QtLinkButton::\_menuitems (C++ member), 512, 1030, 1702, 1799  
 sh::ui::qt::QtLinkButton::menuItemSelected (C++ function), 512, 1030, 1702, 1799  
 sh::ui::qt::QtLinkButton::QtLinkButton (C++ function), 512, 1030, 1702, 1799  
 sh::ui::qt::QtLinkButton::setMenu (C++ function), 512, 1030, 1702, 1799  
 sh::ui::qt::QtLinkButton::setSelectedMenuItem (C++ function), 512, 1030, 1702, 1799  
 sh::ui::qt::QtLinkButton::setSizeByFactor (C++ function), 512, 1030, 1702, 1799  
 sh::ui::qt::QtLogViewDialog (C++ class), 512, 1030, 1702, 1799  
 sh::ui::qt::QtLogViewDialog::\_reloadLog (C++ function), 513, 1031, 1703, 1800  
 sh::ui::qt::QtLogViewDialog::~QtLogViewDialog (C++ function), 512, 1031, 1703, 1800  
 sh::ui::qt::QtLogViewDialog::close (C++ function), 513, 1031, 1703, 1800  
 sh::ui::qt::QtLogViewDialog::dialogId (C++ function), 512, 1031, 1703, 1800  
 sh::ui::qt::QtLogViewDialog::headtext (C++ function), 512, 1031, 1703, 1800  
 sh::ui::qt::QtLogViewDialog::isInitd (C++ function), 513, 1031, 1703, 1800  
 sh::ui::qt::QtLogViewDialog::manager (C++ function), 513, 1031, 1703, 1800  
 sh::ui::qt::QtLogViewDialog::minimumSeverity (C++ function), 512, 1031, 1703, 1800  
 sh::ui::qt::QtLogViewDialog::on\_btnClose\_clicked (C++ function), 513, 1032, 1704, 1801  
 sh::ui::qt::QtLogViewDialog::on\_btnSaveToFile\_clicked (C++ function), 513, 1032, 1704, 1801  
 sh::ui::qt::QtLogViewDialog::on\_btnSeverity\_clicked (C++ function), 513, 1032, 1704, 1801  
 sh::ui::qt::QtLogViewDialog::QtLogViewDialog (C++ function), 512, 1031, 1703, 1800  
 sh::ui::qt::QtLogViewDialog::setMinimumSeverity (C++ function), 513, 1031, 1703, 1800  
 sh::ui::qt::QtLogViewDialog::subheadtxt (C++ function), 512, 1031, 1703, 1800  
 sh::ui::qt::QtLogViewDialog::ui (C++ member), 513, 1032, 1704, 1801  
 sh::ui::qt::QtLogViewDialog::waitClosed (C++ function), 513, 1031, 1703, 1800  
 sh::ui::qt::QtLogViewDialog::wasAccepted (C++ function), 513, 1031, 1703, 1800  
 sh::ui::qt::QtLogViewDialog::wasClosed (C++ function), 513, 1031, 1703, 1800  
 sh::ui::qt::QtMainWindow (C++ class), 514, 1032, 1704, 1801  
 sh::ui::qt::QtMainWindow::\_addPermanentActionToToolBar (C++ function), 517, 1036, 1708, 1805  
 sh::ui::qt::QtMainWindow::\_closeDirectly (C++ function), 516, 517, 1035, 1707, 1804  
 sh::ui::qt::QtMainWindow::\_currentDirectory (C++ member), 518, 1036, 1708, 1805  
 sh::ui::qt::QtMainWindow::\_detailpanel\_resized (C++ function), 517, 1036, 1708, 1805  
 sh::ui::qt::QtMainWindow::\_dialogManager (C++ member), 518, 1036, 1708, 1805  
 sh::ui::qt::QtMainWindow::\_initialize (C++ function), 515, 1034, 1706, 1803  
 sh::ui::qt::QtMainWindow::\_jumpToIndex (C++ function), 517, 1036, 1708, 1805  
 sh::ui::qt::QtMainWindow::\_jumpToNode (C++ function), 517, 1036, 1708, 1805  
 sh::ui::qt::QtMainWindow::\_myorientation (C++ member), 518, 1036, 1708, 1805  
 sh::ui::qt::QtMainWindow::\_newToStatusbar\_helper (C++ function), 517, 1036, 1708, 1805

sh::ui::qt::QtMainWindow::\_qtMainWindow (C++ function), 514, 1032, 1704, 1801  
 (C++ member), 518, 1037, 1709, 1806  
 sh::ui::qt::QtMainWindow::\_refresh\_detailthumbbr (C++ function), 516, 1035, 1707, 1804  
 (C++ function), 517, 1036, 1708, 1805  
 sh::ui::qt::QtMainWindow::\_refreshToolbar (C++ function), 514, 1032, 1704, 1801  
 (C++ function), 517, 1036, 1708, 1805  
 sh::ui::qt::QtMainWindow::\_request\_detailthumbbr (C++ function), 515, 1034, 1706, 1803  
 (C++ member), 518, 1036, 1708, 1805  
 sh::ui::qt::QtMainWindow::\_resizedetailpanel (C++ function), 517, 1036, 1708, 1805  
 (C++ function), 517, 1036, 1708, 1805  
 sh::ui::qt::QtMainWindow::\_resizethumbnail (C++ function), 514, 1032, 1704, 1801  
 (C++ function), 517, 1036, 1708, 1805  
 sh::ui::qt::QtMainWindow::\_setupGlobalShortcut (C++ function), 515, 1034, 1706, 1803  
 (C++ function), 517, 1036, 1708, 1805  
 sh::ui::qt::QtMainWindow::\_setupGlobalShortcut (C++ function), 515, 1034, 1706, 1803  
 (C++ function), 517, 1036, 1708, 1805  
 sh::ui::qt::QtMainWindow::\_statusbar\_childs (C++ function), 514, 1032, 1704, 1801  
 (C++ member), 518, 1036, 1708, 1805  
 sh::ui::qt::QtMainWindow::\_statusbar\_fullheight (C++ function), 517, 1035, 1707, 1804  
 (C++ member), 518, 1036, 1708, 1805  
 sh::ui::qt::QtMainWindow::\_statusbar\_targetheight (C++ function), 517, 1035, 1707, 1804  
 (C++ member), 518, 1036, 1708, 1805  
 sh::ui::qt::QtMainWindow::\_statusbar\_timer (C++ function), 517, 1035, 1707, 1804  
 (C++ member), 518, 1036, 1708, 1805  
 sh::ui::qt::QtMainWindow::\_thumbnail\_resized (C++ function), 517, 1036, 1708, 1805  
 (C++ function), 517, 1036, 1708, 1805  
 sh::ui::qt::QtMainWindow::\_thumbnailwidth (C++ function), 514, 1033, 1705, 1802  
 (C++ member), 518, 1036, 1708, 1805  
 sh::ui::qt::QtMainWindow::\_toolbarLeftRightSplit (C++ function), 514, 1033, 1705, 1802  
 (C++ member), 518, 1036, 1708, 1805  
 sh::ui::qt::QtMainWindow::\_uistyle (C++ member), 518, 1036, 1708, 1805  
 sh::ui::qt::QtMainWindow::~QtMainWindow (C++ function), 514, 1032, 1704, 1801  
 sh::ui::qt::QtMainWindow::actionGroupsActions (C++ function), 514, 1032, 1704, 1801  
 (C++ member), 518, 1036, 1708, 1805  
 sh::ui::qt::QtMainWindow::addFileView (C++ function), 514, 1032, 1704, 1801  
 (C++ function), 514, 1032, 1704, 1801  
 sh::ui::qt::QtMainWindow::addInfoPanel (C++ function), 514, 516, 1032, 1035, 1704, 1707, 1801, 1804  
 sh::ui::qt::QtMainWindow::closeApp (C++ function), 517, 1035, 1707, 1804  
 sh::ui::qt::QtMainWindow::createActionExecutionInfoForQtMainWindow::onCurrentDirectoryChanged (C++ function), 515, 1034, 1705, 1802  
 sh::ui::qt::QtMainWindow::currentDirectory (C++ function), 514, 1032, 1704, 1801  
 sh::ui::qt::QtMainWindow::currentFileView (C++ function), 514, 1032, 1704, 1801  
 sh::ui::qt::QtMainWindow::currentProfiles (C++ function), 516, 1034, 1706, 1803  
 sh::ui::qt::QtMainWindow::detailPanelPosition (C++ function), 515, 1033, 1705, 1802  
 sh::ui::qt::QtMainWindow::dialogManager (C++ function), 514, 1032, 1704, 1801  
 sh::ui::qt::QtMainWindow::fileDetailsPanelVisible (C++ function), 516, 1035, 1707, 1804  
 sh::ui::qt::QtMainWindow::fileViewCount (C++ function), 514, 1032, 1704, 1801  
 sh::ui::qt::QtMainWindow::fileViewsReloadAll (C++ function), 515, 1034, 1706, 1803  
 sh::ui::qt::QtMainWindow::fspanel (C++ member), 518, 1036, 1708, 1805  
 sh::ui::qt::QtMainWindow::getFileView (C++ function), 514, 1032, 1704, 1801  
 sh::ui::qt::QtMainWindow::handleCloseAppRejected (C++ function), 515, 1034, 1706, 1803  
 sh::ui::qt::QtMainWindow::handleClosed (C++ function), 515, 1034, 1706, 1803  
 sh::ui::qt::QtMainWindow::initialize (C++ function), 514, 1032, 1704, 1801  
 sh::ui::qt::QtMainWindow::isCloseable (C++ function), 517, 1035, 1707, 1804  
 sh::ui::qt::QtMainWindow::isClosing (C++ function), 517, 1035, 1707, 1804  
 sh::ui::qt::QtMainWindow::isReady (C++ function), 517, 1035, 1707, 1804  
 sh::ui::qt::QtMainWindow::jumpbar (C++ member), 518, 1036, 1708, 1805  
 sh::ui::qt::QtMainWindow::jumpbarIsTextMode (C++ function), 514, 1033, 1705, 1802  
 sh::ui::qt::QtMainWindow::jumpbarSetButtonMode (C++ function), 514, 1033, 1705, 1802  
 sh::ui::qt::QtMainWindow::jumpbarSetTextMode (C++ function), 514, 1033, 1705, 1802  
 sh::ui::qt::QtMainWindow::jumpToEurl (C++ function), 516, 1034, 1706, 1803  
 sh::ui::qt::QtMainWindow::jumpToNode (C++ function), 514, 1032, 1704, 1801  
 sh::ui::qt::QtMainWindow::mainWindow (C++ function), 517, 1035, 1707, 1804  
 sh::ui::qt::QtMainWindow::MyFlexibleLabel (C++ class), 518, 519, 1037, 1709, 1806  
 sh::ui::qt::QtMainWindow::MyFlexibleLabel::MyFlexibleLabel (C++ function), 519, 1037, 1709, 1806  
 sh::ui::qt::QtMainWindow::MyFlexibleLabel::sizeHint (C++ function), 519, 1037, 1709, 1806  
 sh::ui::qt::QtMainWindow::onCurrentDirectoryChanged (C++ function), 516, 1034, 1706, 1803  
 sh::ui::qt::QtMainWindow::onCurrentFileViewChanged (C++ function), 516, 1034, 1706, 1803  
 sh::ui::qt::QtMainWindow::onCurrentProfileChanged (C++ function), 516, 1034, 1706, 1803  
 sh::ui::qt::QtMainWindow::onFileViewCountChanged (C++ function), 516, 1034, 1706, 1803  
 sh::ui::qt::QtMainWindow::onFileViewOptionsChanged (C++ function), 516, 1034, 1706, 1803  
 sh::ui::qt::QtMainWindow::onGlobalViewOptionsChanged (C++ function), 516, 1034, 1706, 1803



(C++ function), 516, 1034, 1706, 1803  
 sh::ui::qt::QtMainWindow::QtMainWindow (C++ function), 514, 1032, 1704, 1801  
 sh::ui::qt::QtMainWindow::reloadProfile (C++ function), 516, 1034, 1706, 1803  
 sh::ui::qt::QtMainWindow::removeFileViews (C++ function), 514, 1032, 1704, 1801  
 sh::ui::qt::QtMainWindow::requestDetailThumbnail (C++ function), 517, 1036, 1708, 1805  
 sh::ui::qt::QtMainWindow::runsHeadless (C++ function), 517, 1035, 1707, 1804  
 sh::ui::qt::QtMainWindow::setCurrentProfile (C++ function), 516, 1034, 1706, 1803  
 sh::ui::qt::QtMainWindow::setDetailPanelPosition (C++ function), 515, 1033, 1705, 1802  
 sh::ui::qt::QtMainWindow::setFileDetailsPanelVisible (C++ function), 514, 1032, 1704, 1801  
 sh::ui::qt::QtMainWindow::setMainWindow (C++ function), 517, 1035, 1707, 1804  
 sh::ui::qt::QtMainWindow::setStatusBarItemVisibility (C++ function), 517, 1036, 1708, 1805  
 sh::ui::qt::QtMainWindow::setTitlePattern (C++ function), 514, 1032, 1704, 1801  
 sh::ui::qt::QtMainWindow::setToolBarPosition (C++ function), 515, 1033, 1705, 1802  
 sh::ui::qt::QtMainWindow::setTreeSticky (C++ function), 516, 1035, 1706, 1803  
 sh::ui::qt::QtMainWindow::setTreeVisible (C++ function), 514, 1032, 1704, 1801  
 sh::ui::qt::QtMainWindow::showAboutDialog (C++ function), 515, 1033, 1705, 1802  
 sh::ui::qt::QtMainWindow::showErrorPanel (C++ function), 516, 1035, 1707, 1804  
 sh::ui::qt::QtMainWindow::showFilePropertiesDialog (C++ function), 515, 1033, 1705, 1802  
 sh::ui::qt::QtMainWindow::showLogViewDialog (C++ function), 515, 1033, 1705, 1802  
 sh::ui::qt::QtMainWindow::showManageBookmarksDialog (C++ function), 515, 1033, 1705, 1802  
 sh::ui::qt::QtMainWindow::showManageProfilesDialog (C++ function), 515, 1033, 1705, 1802  
 sh::ui::qt::QtMainWindow::showOpenWithDialog (C++ function), 515, 1033, 1705, 1802  
 sh::ui::qt::QtMainWindow::showStoreProfileDialog (C++ function), 515, 1033, 1705, 1802  
 sh::ui::qt::QtMainWindow::showTuningDialog (C++ function), 515, 1033, 1705, 1802  
 sh::ui::qt::QtMainWindow::slot\_detailbar\_moved (C++ function), 518, 1037, 1709, 1806  
 sh::ui::qt::QtMainWindow::slot\_statusbar\_hidden (C++ function), 518, 1037, 1709, 1806  
 sh::ui::qt::QtMainWindow::slot\_toolbar\_moved (C++ function), 518, 1037, 1709, 1806  
 sh::ui::qt::QtMainWindow::slot\_treeview\_shallot\_expanded (C++ function), 518, 1037, 1709, 1806  
 sh::ui::qt::QtMainWindow::titlePattern (C++ function), 516, 1034, 1706, 1803  
 sh::ui::qt::QtMainWindow::toolbarButtonHandlers (C++ member), 518, 1036, 1708, 1805  
 sh::ui::qt::QtMainWindow::toolbarPosition (C++ function), 515, 1033, 1705, 1802  
 sh::ui::qt::QtMainWindow::treemodel (C++ member), 518, 1036, 1708, 1805  
 sh::ui::qt::QtMainWindow::treeSticky (C++ function), 516, 1035, 1707, 1804  
 sh::ui::qt::QtMainWindow::treeVisible (C++ function), 516, 1034, 1706, 1803  
 sh::ui::qt::QtMainWindow::tryCreateMainWindow (C++ function), 517, 1035, 1707, 1804  
 sh::ui::qt::QtMainWindow::ui (C++ member), 518, 1036, 1708, 1805  
 sh::ui::qt::QtMainWindow::uiMode (C++ function), 517, 1035, 1707, 1804  
 sh::ui::qt::QtMainWindow::uiStyle (C++ function), 515, 1034, 1706, 1803  
 sh::ui::qt::QtManageBookmarksDialog (C++ class), 519, 1037, 1709, 1806  
 sh::ui::qt::QtManageBookmarksDialog::\_refresh\_value (C++ function), 520, 1038, 1710, 1807  
 sh::ui::qt::QtManageBookmarksDialog::\_selectbookmark (C++ function), 520, 1038, 1710, 1807  
 sh::ui::qt::QtManageBookmarksDialog::\_selectfolder (C++ function), 520, 1038, 1710, 1807  
 sh::ui::qt::QtManageBookmarksDialog::~QtManageBookmarksDialog (C++ function), 519, 1037, 1709, 1806  
 sh::ui::qt::QtManageBookmarksDialog::close (C++ function), 519, 1037, 1709, 1806  
 sh::ui::qt::QtManageBookmarksDialog::dialogId (C++ function), 519, 1037, 1709, 1806  
 sh::ui::qt::QtManageBookmarksDialog::isInitd (C++ function), 519, 1037, 1709, 1806  
 sh::ui::qt::QtManageBookmarksDialog::manager (C++ function), 519, 1038, 1710, 1807  
 sh::ui::qt::QtManageBookmarksDialog::on\_btnClose\_clicked (C++ function), 520, 1038, 1710, 1807  
 sh::ui::qt::QtManageBookmarksDialog::on\_btnDelete\_clicked (C++ function), 520, 1038, 1710, 1807  
 sh::ui::qt::QtManageBookmarksDialog::on\_btnDown\_clicked (C++ function), 520, 1038, 1710, 1807  
 sh::ui::qt::QtManageBookmarksDialog::on\_btnMove\_clicked (C++ function), 520, 1038, 1710, 1807  
 sh::ui::qt::QtManageBookmarksDialog::on\_btnNew\_clicked (C++ function), 520, 1038, 1710, 1807  
 sh::ui::qt::QtManageBookmarksDialog::on\_btnStore\_clicked (C++ function), 520, 1038, 1710, 1807  
 sh::ui::qt::QtManageBookmarksDialog::on\_btnUp\_clicked (C++ function), 520, 1038, 1710, 1807  
 sh::ui::qt::QtManageBookmarksDialog::on\_edtLabel\_textChanged (C++ function), 520, 1038, 1710, 1807

(C++ function), 520, 1038, 1710, 1807  
 sh::ui::qt::QtManageBookmarksDialog::on\_editLocationBtn\_QtManageProfilesDialog::waitClosed  
 (C++ function), 520, 1038, 1710, 1807  
 sh::ui::qt::QtManageBookmarksDialog::on\_treeWidgetqt::QtManageProfilesDialog::wasAccepted  
 (C++ function), 520, 1038, 1710, 1807  
 sh::ui::qt::QtManageBookmarksDialog::QtManageBookmarksDialog::QtManageProfilesDialog::wasClosed  
 (C++ function), 519, 1037, 1709, 1806  
 sh::ui::qt::QtManageBookmarksDialog::readBookmarks::QtOpenWithDialog (C++ class),  
 (C++ function), 520, 1038, 1710, 1807  
 sh::ui::qt::QtManageBookmarksDialog::ui sh::ui::qt::QtOpenWithDialog::\_chosenMethod  
 (C++ member), 520, 1038, 1710, 1807  
 sh::ui::qt::QtManageBookmarksDialog::waitClosed::qt::QtOpenWithDialog::\_listmodel  
 (C++ function), 519, 1037, 1709, 1806  
 sh::ui::qt::QtManageBookmarksDialog::wasAccepted::qt::QtOpenWithDialog::\_rememberfordirectory  
 (C++ function), 519, 1037, 1709, 1806  
 sh::ui::qt::QtManageBookmarksDialog::wasClosed::qt::QtOpenWithDialog::~QtOpenWithDialog  
 (C++ function), 519, 1038, 1710, 1807  
 sh::ui::qt::QtManageProfilesDialog (C++ sh::ui::qt::QtOpenWithDialog::chosenMethod  
 class), 520, 1038, 1710, 1807  
 sh::ui::qt::QtManageProfilesDialog::\_createNodes::qt::QtOpenWithDialog::close  
 (C++ function), 521, 1039, 1711, 1808  
 sh::ui::qt::QtManageProfilesDialog::\_createProfilesList::qt::QtOpenWithDialog::dialogId  
 (C++ function), 521, 1039, 1711, 1808  
 sh::ui::qt::QtManageProfilesDialog::\_createSettingsList::qt::QtOpenWithDialog::isInited  
 (C++ function), 521, 1039, 1711, 1808  
 sh::ui::qt::QtManageProfilesDialog::\_currentNode::qt::QtOpenWithDialog::manager  
 (C++ member), 521, 1039, 1711, 1808  
 sh::ui::qt::QtManageProfilesDialog::\_model::ui::qt::QtOpenWithDialog::node (C++  
 (C++ member), 521, 1039, 1711, 1808  
 sh::ui::qt::QtManageProfilesDialog::\_nodes::ui::qt::QtOpenWithDialog::on\_btnAnotherAncestor  
 (C++ member), 521, 1039, 1711, 1808  
 sh::ui::qt::QtManageProfilesDialog::~QtManageProfilesDialog::on\_btnCancel\_clicked  
 (C++ function), 520, 1039, 1711, 1808  
 sh::ui::qt::QtManageProfilesDialog::close::ui::qt::QtOpenWithDialog::on\_btnOk\_clicked  
 (C++ function), 521, 1039, 1711, 1808  
 sh::ui::qt::QtManageProfilesDialog::dialogId::ui::qt::QtOpenWithDialog::on\_btnSelectOther\_cl  
 (C++ function), 520, 1039, 1711, 1808  
 sh::ui::qt::QtManageProfilesDialog::isInited::ui::qt::QtOpenWithDialog::on\_chkRememberDir\_toq  
 (C++ function), 520, 1039, 1711, 1808  
 sh::ui::qt::QtManageProfilesDialog::manager::ui::qt::QtOpenWithDialog::on\_listView\_activated  
 (C++ function), 521, 1039, 1711, 1808  
 sh::ui::qt::QtManageProfilesDialog::on\_btnClose\_clicked::qt::QtOpenWithDialog::openMethods  
 (C++ function), 521, 1040, 1712, 1809  
 sh::ui::qt::QtManageProfilesDialog::on\_btnDelete\_clicked::qt::QtOpenWithDialog::QtOpenWithDialog  
 (C++ function), 521, 1040, 1712, 1809  
 sh::ui::qt::QtManageProfilesDialog::on\_btnDeleteProfileQtOpenWithDialog::rememberForDirectory  
 (C++ function), 521, 1040, 1712, 1809  
 sh::ui::qt::QtManageProfilesDialog::on\_comboBox::qt::QtOpenWithDialog::rememberForFile  
 (C++ function), 521, 1040, 1712, 1809  
 sh::ui::qt::QtManageProfilesDialog::QtManageProfilesDialog::QtOpenWithDialog::rememberForMimetype  
 (C++ function), 520, 1039, 1711, 1808  
 sh::ui::qt::QtManageProfilesDialog::slots::ui::qt::QtOpenWithDialog::setProgramList  
 (C++ function), 521, 1040, 1712, 1809  
 sh::ui::qt::QtManageProfilesDialog::ui sh::ui::qt::QtOpenWithDialog::ui (C++

*member*), 523, 1041, 1713, 1810 (C++ function), 525, 1043, 1715, 1812  
 sh::ui::qt::QtOpenWithDialog::waitClosed sh::ui::qt::QtSearchPanelConfiguration::addSpacer  
 (C++ function), 522, 1040, 1712, 1809 (C++ function), 525, 1043, 1715, 1812  
 sh::ui::qt::QtOpenWithDialog::wasAccepted sh::ui::qt::QtSearchPanelConfiguration::addTextEditor  
 (C++ function), 522, 1040, 1712, 1809 (C++ function), 525, 1043, 1715, 1812  
 sh::ui::qt::QtOpenWithDialog::wasClosed sh::ui::qt::QtSearchPanelConfiguration::editors  
 (C++ function), 522, 1041, 1713, 1810 (C++ member), 525, 1043, 1715, 1812  
 sh::ui::qt::QtOpenWithDialogModel (C++ sh::ui::qt::QtSearchPanelConfiguration::getEditorAt  
 class), 523, 1041, 1713, 1810 (C++ function), 525, 1043, 1715, 1812  
 sh::ui::qt::QtOpenWithDialogModel::\_methods sh::ui::qt::QtSearchPanelConfiguration::myqobject  
 (C++ member), 523, 1042, 1714, 1811 (C++ member), 525, 1043, 1715, 1812  
 sh::ui::qt::QtOpenWithDialogModel::data sh::ui::qt::QtSearchPanelConfiguration::onDestroyed  
 (C++ function), 523, 1041, 1713, 1810 (C++ function), 525, 1043, 1715, 1812  
 sh::ui::qt::QtOpenWithDialogModel::QtOpenWithDialogModel sh::ui::qt::QtSearchPanelConfiguration::owner  
 (C++ function), 523, 1041, 1713, 1810 (C++ member), 525, 1043, 1715, 1812  
 sh::ui::qt::QtSearchPanel (C++ class), 523, sh::ui::qt::QtSearchPanelConfiguration::QtSearchPanel  
 1042, 1714, 1811 (C++ function), 525, 1043, 1715, 1812  
 sh::ui::qt::QtSearchPanel::~~QtSearchPanel sh::ui::qt::QtSearchPanelDateTimeEditor  
 (C++ function), 524, 1042, 1714, 1811 (C++ class), 526, 1043, 1715, 1812  
 sh::ui::qt::QtSearchPanel::QtSearchPanel sh::ui::qt::QtSearchPanelDateTimeEditor::datetime  
 (C++ function), 524, 1042, 1714, 1811 (C++ function), 526, 1044, 1716, 1813  
 sh::ui::qt::QtSearchPanelAbstractEditor sh::ui::qt::QtSearchPanelDateTimeEditor::isWidgetEnabled  
 (C++ class), 524, 1042, 1714, 1811 (C++ function), 526, 1044, 1716, 1813  
 sh::ui::qt::QtSearchPanelAbstractEditor::show sh::ui::qt::QtSearchPanelDateTimeEditor::QtSearchPanel  
 (C++ function), 524, 1042, 1714, 1811 (C++ function), 526, 1044, 1716, 1813  
 sh::ui::qt::QtSearchPanelAbstractEditor::show sh::ui::qt::QtSearchPanelDateTimeEditor::setDatetime  
 (C++ function), 524, 1042, 1714, 1811 (C++ function), 526, 1044, 1716, 1813  
 sh::ui::qt::QtSearchPanelAbstractEditor::show sh::ui::qt::QtSearchPanelDateTimeEditor::setWidgetEnabled  
 (C++ function), 524, 1042, 1714, 1811 (C++ function), 526, 1044, 1716, 1813  
 sh::ui::qt::QtSearchPanelButton (C++ sh::ui::qt::QtSearchPanelLabelEditor  
 class), 524, 1042, 1714, 1811 (C++ class), 526, 1044, 1716, 1813  
 sh::ui::qt::QtSearchPanelButton::menuItemSelected sh::ui::qt::QtSearchPanelLabelEditor::focusProxyEditor  
 (C++ function), 525, 1043, 1715, 1812 (C++ function), 526, 1044, 1716, 1813  
 sh::ui::qt::QtSearchPanelButton::QtSearchPanelButton sh::ui::qt::QtSearchPanelLabelEditor::isWidgetEnabled  
 (C++ function), 524, 1042, 1714, 1811 (C++ function), 526, 1044, 1716, 1813  
 sh::ui::qt::QtSearchPanelButton::setButtonText sh::ui::qt::QtSearchPanelLabelEditor::QtSearchPanel  
 (C++ function), 524, 1042, 1714, 1811 (C++ function), 526, 1044, 1716, 1813  
 sh::ui::qt::QtSearchPanelButton::setMenu sh::ui::qt::QtSearchPanelLabelEditor::setFocusProxyEditor  
 (C++ function), 524, 1042, 1714, 1811 (C++ function), 526, 1044, 1716, 1813  
 sh::ui::qt::QtSearchPanelButton::setMenuSelection sh::ui::qt::QtSearchPanelLabelEditor::setTextContent  
 (C++ function), 524, 1042, 1714, 1811 (C++ function), 526, 1044, 1716, 1813  
 sh::ui::qt::QtSearchPanelButton::setSelectedMenuItem sh::ui::qt::QtSearchPanelLabelEditor::setWidgetEnabled  
 (C++ function), 524, 1042, 1714, 1811 (C++ function), 526, 1044, 1716, 1813  
 sh::ui::qt::QtSearchPanelButton::setSizeByFactor sh::ui::qt::QtSearchPanelLabelEditor::textContent  
 (C++ function), 524, 1042, 1714, 1811 (C++ function), 526, 1044, 1716, 1813  
 sh::ui::qt::QtSearchPanelConfiguration sh::ui::qt::QtSearchPanelSpacerEditor  
 (C++ class), 525, 1043, 1715, 1812 (C++ class), 526, 1044, 1716, 1813  
 sh::ui::qt::QtSearchPanelConfiguration::addAction sh::ui::qt::QtSearchPanelSpacerEditor::isWidgetEnabled  
 (C++ function), 525, 1043, 1715, 1812 (C++ function), 527, 1044, 1716, 1813  
 sh::ui::qt::QtSearchPanelConfiguration::addDateTimeEditor sh::ui::qt::QtSearchPanelSpacerEditor::QtSearchPanel  
 (C++ function), 525, 1043, 1715, 1812 (C++ function), 527, 1044, 1716, 1813  
 sh::ui::qt::QtSearchPanelConfiguration::addLabel sh::ui::qt::QtSearchPanelSpacerEditor::setWidgetEnabled  
 (C++ function), 525, 1043, 1715, 1812 (C++ function), 527, 1044, 1716, 1813  
 sh::ui::qt::QtSearchPanelConfiguration::addMenuButton sh::ui::qt::QtSearchPanelTextEditor

(C++ class), 527, 1044, 1716, 1813 (C++ function), 528, 1046, 1718, 1815  
sh::ui::qt::QtSearchPanelTextEditor::isWidgetEnabledQtStoreProfileDialog::checkedSettings  
(C++ function), 527, 1045, 1717, 1814 (C++ function), 528, 1046, 1718, 1815  
sh::ui::qt::QtSearchPanelTextEditor::plashholderDescQtStoreProfileDialog::close  
(C++ function), 527, 1045, 1717, 1814 (C++ function), 529, 1046, 1718, 1815  
sh::ui::qt::QtSearchPanelTextEditor::QtSearchPanelTextEditorQtStoreProfileDialog::dialogId  
(C++ function), 527, 1045, 1717, 1814 (C++ function), 528, 1046, 1718, 1815  
sh::ui::qt::QtSearchPanelTextEditor::setPlaceholderDescQtStoreProfileDialog::eurl  
(C++ function), 527, 1045, 1717, 1814 (C++ function), 528, 1046, 1718, 1815  
sh::ui::qt::QtSearchPanelTextEditor::setTextContentQtStoreProfileDialog::hasGlobal  
(C++ function), 527, 1045, 1717, 1814 (C++ function), 528, 1046, 1718, 1815  
sh::ui::qt::QtSearchPanelTextEditor::setWidgetEnabledQtStoreProfileDialog::inheritsFrom  
(C++ function), 527, 1045, 1717, 1814 (C++ function), 528, 1046, 1718, 1815  
sh::ui::qt::QtSearchPanelTextEditor::textContentQt::QtStoreProfileDialog::isInitd  
(C++ function), 527, 1045, 1717, 1814 (C++ function), 528, 1046, 1718, 1815  
sh::ui::qt::QtSettingUIFrame (C++ class), sh::ui::qt::QtStoreProfileDialog::manager  
527, 1045, 1717, 1814 (C++ function), 529, 1046, 1718, 1815  
sh::ui::qt::QtSettingUIFrame::\_additionalCheckBoxQt::QtStoreProfileDialog::on\_btn22\_clicked  
(C++ member), 528, 1045, 1717, 1814 (C++ function), 529, 1047, 1719, 1816  
sh::ui::qt::QtSettingUIFrame::\_additionalCheckBoxQt::QtStoreProfileDialog::on\_btnAddProf\_cl  
(C++ member), 528, 1045, 1717, 1814 (C++ function), 529, 1047, 1719, 1816  
sh::ui::qt::QtSettingUIFrame::\_additionalCheckBoxQt::QtStoreProfileDialog::on\_btnGlobal\_cli  
(C++ member), 528, 1045, 1717, 1814 (C++ function), 529, 1047, 1719, 1816  
sh::ui::qt::QtSettingUIFrame::\_additionalCheckBoxQt::QtStoreProfileDialog::on\_toolButton\_2\_  
(C++ member), 528, 1045, 1717, 1814 (C++ function), 529, 1047, 1719, 1816  
sh::ui::qt::QtSettingUIFrame::\_checkbox sh::ui::qt::QtStoreProfileDialog::on\_toolButton\_cl  
(C++ member), 528, 1045, 1717, 1814 (C++ function), 529, 1047, 1719, 1816  
sh::ui::qt::QtSettingUIFrame::additionalCheckValueQt::QtStoreProfileDialog::profileName  
(C++ function), 527, 1045, 1717, 1814 (C++ function), 528, 1046, 1718, 1815  
sh::ui::qt::QtSettingUIFrame::isChecked sh::ui::qt::QtStoreProfileDialog::QtStoreProfileDia  
(C++ function), 527, 1045, 1717, 1814 (C++ function), 528, 1046, 1718, 1815  
sh::ui::qt::QtSettingUIFrame::QtSettingUIFrameQt::QtStoreProfileDialog::ui  
(C++ function), 527, 1045, 1717, 1814 (C++ member), 529, 1047, 1719, 1816  
sh::ui::qt::QtSettingUIFrame::setAdditionalCheckBoxQt::QtStoreProfileDialog::waitClosed  
(C++ function), 527, 1045, 1717, 1814 (C++ function), 528, 1046, 1718, 1815  
sh::ui::qt::QtSettingUIFrame::setCheckedsh::ui::qt::QtStoreProfileDialog::wasAccepted  
(C++ function), 527, 1045, 1717, 1814 (C++ function), 528, 1046, 1718, 1815  
sh::ui::qt::QtStoreProfileDialog (C++ sh::ui::qt::QtStoreProfileDialog::wasClosed  
class), 528, 1045, 1717, 1814 (C++ function), 529, 1046, 1718, 1815  
sh::ui::qt::QtStoreProfileDialog::\_globalsh::ui::qt::QtStoreProfileDialog::withSubDirectorie  
(C++ member), 529, 1047, 1719, 1816 (C++ function), 528, 1046, 1718, 1815  
sh::ui::qt::QtStoreProfileDialog::\_headerLabelsQt::QtToolBarButton (C++ class),  
(C++ member), 529, 1047, 1719, 1816 530, 1047, 1719, 1816  
sh::ui::qt::QtStoreProfileDialog::\_inheritsProfileNameQtToolBarButton::\_arrowhovered  
(C++ member), 529, 1047, 1719, 1816 (C++ member), 531, 1048, 1720, 1817  
sh::ui::qt::QtStoreProfileDialog::\_levelsh::ui::qt::QtToolBarButton::\_calcDims  
(C++ member), 529, 1047, 1719, 1816 (C++ function), 531, 1048, 1720, 1817  
sh::ui::qt::QtStoreProfileDialog::\_setglobalui::qt::QtToolBarButton::\_clickAction  
(C++ function), 529, 1047, 1719, 1816 (C++ member), 531, 1048, 1720, 1817  
sh::ui::qt::QtStoreProfileDialog::\_settingsLevelQt::QtToolBarButton::\_drawIconOnly  
(C++ function), 529, 1047, 1719, 1816 (C++ member), 531, 1048, 1720, 1817  
sh::ui::qt::QtStoreProfileDialog::\_widgetsh::ui::qt::QtToolBarButton::\_drawmnemonics  
(C++ member), 529, 1047, 1719, 1816 (C++ member), 532, 1049, 1721, 1818  
sh::ui::qt::QtStoreProfileDialog::~QtStoreProfileDialogQtToolBarButton::\_hovered



(C++ member), 531, 1048, 1720, 1817  
 sh::ui::qt::QtToolBarButton::\_icon (C++ member), 531, 1048, 1720, 1817  
 sh::ui::qt::QtToolBarButton::\_menu (C++ member), 531, 1048, 1720, 1817  
 sh::ui::qt::QtToolBarButton::\_text (C++ member), 531, 1048, 1720, 1817  
 sh::ui::qt::QtToolBarButton::\_text1 (C++ member), 531, 1048, 1720, 1817  
 sh::ui::qt::QtToolBarButton::\_text2 (C++ member), 531, 1048, 1720, 1817  
 sh::ui::qt::QtToolBarButton::\_textWithoutMnemonic (C++ member), 531, 1048, 1720, 1817  
 sh::ui::qt::QtToolBarButton::\_textWithoutMnemonic2 (C++ member), 531, 1048, 1720, 1817  
 sh::ui::qt::QtToolBarButton::arrow (C++ member), 532, 1049, 1721, 1818  
 sh::ui::qt::QtToolBarButton::buttonIcon (C++ function), 530, 1048, 1720, 1817  
 sh::ui::qt::QtToolBarButton::buttonText (C++ function), 530, 1048, 1720, 1817  
 sh::ui::qt::QtToolBarButton::computeArrowAndDividerLocation (C++ function), 531, 1048, 1720, 1817  
 sh::ui::qt::QtToolBarButton::eh (C++ member), 531, 1049, 1721, 1818  
 sh::ui::qt::QtToolBarButton::ew (C++ member), 531, 1049, 1721, 1818  
 sh::ui::qt::QtToolBarButton::exp1 (C++ member), 531, 1049, 1721, 1818  
 sh::ui::qt::QtToolBarButton::exp2 (C++ member), 531, 1049, 1721, 1818  
 sh::ui::qt::QtToolBarButton::expy1 (C++ member), 531, 1049, 1721, 1818  
 sh::ui::qt::QtToolBarButton::expy2 (C++ member), 531, 1049, 1721, 1818  
 sh::ui::qt::QtToolBarButton::hasExpanders (C++ function), 530, 1048, 1720, 1817  
 sh::ui::qt::QtToolBarButton::icon (C++ function), 530, 1047, 1719, 1816  
 sh::ui::qt::QtToolBarButton::iedim (C++ member), 531, 1049, 1721, 1818  
 sh::ui::qt::QtToolBarButton::ih (C++ member), 531, 1049, 1721, 1818  
 sh::ui::qt::QtToolBarButton::isButtonEnabled (C++ function), 530, 1048, 1720, 1817  
 sh::ui::qt::QtToolBarButton::iw (C++ member), 531, 1049, 1721, 1818  
 sh::ui::qt::QtToolBarButton::location (C++ member), 531, 1048, 1720, 1817  
 sh::ui::qt::QtToolBarButton::opad (C++ member), 531, 1049, 1721, 1818  
 sh::ui::qt::QtToolBarButton::openMenu (C++ function), 530, 1048, 1720, 1817  
 sh::ui::qt::QtToolBarButton::Position (C++ enum), 530, 1047, 1719, 1816  
 sh::ui::qt::QtToolBarButton::Position::BOTTOM (C++ enumerator), 530, 1047, 1719, 1816  
 sh::ui::qt::QtToolBarButton::Position::LEFT (C++ enumerator), 530, 1047, 1719, 1816  
 sh::ui::qt::QtToolBarButton::Position::RIGHT (C++ enumerator), 530, 1047, 1719, 1816  
 sh::ui::qt::QtToolBarButton::Position::TOP (C++ enumerator), 530, 1047, 1719, 1816  
 sh::ui::qt::QtToolBarButton::QtToolBarButton (C++ function), 530, 1047, 1719, 1816  
 sh::ui::qt::QtToolBarButton::setButtonEnabled (C++ function), 530, 1048, 1720, 1817  
 sh::ui::qt::QtToolBarButton::setButtonIcon (C++ function), 530, 1048, 1720, 1817  
 sh::ui::qt::QtToolBarButton::setButtonText (C++ function), 530, 1047, 1719, 1816  
 sh::ui::qt::QtToolBarButton::setClickAction (C++ function), 530, 1048, 1720, 1817  
 sh::ui::qt::QtToolBarButton::setIcon (C++ function), 530, 1047, 1719, 1816  
 sh::ui::qt::QtToolBarButton::setLocation (C++ function), 530, 1048, 1720, 1817  
 sh::ui::qt::QtToolBarButton::setSubMenu (C++ function), 530, 1048, 1720, 1817  
 sh::ui::qt::QtToolBarButton::setText (C++ function), 530, 1047, 1719, 1816  
 sh::ui::qt::QtToolBarButton::sizeHint (C++ function), 530, 1047, 1719, 1816  
 sh::ui::qt::QtToolBarButton::slot\_clicked (C++ function), 532, 1049, 1721, 1818  
 sh::ui::qt::QtToolBarButton::submenu (C++ function), 530, 1048, 1720, 1817  
 sh::ui::qt::QtToolBarButton::text (C++ function), 530, 1047, 1719, 1816  
 sh::ui::qt::QtToolBarButton::textpad (C++ member), 531, 1049, 1721, 1818  
 sh::ui::qt::QtToolBarButton::th (C++ member), 531, 1049, 1721, 1818  
 sh::ui::qt::QtToolBarButton::thorig (C++ member), 531, 1049, 1721, 1818  
 sh::ui::qt::QtToolBarButton::trigger (C++ function), 530, 1048, 1720, 1817  
 sh::ui::qt::QtToolBarButton::tw1 (C++ member), 531, 1048, 1720, 1817  
 sh::ui::qt::QtToolBarButton::tw2 (C++ member), 531, 1048, 1720, 1817  
 sh::ui::qt::QtToolBarButton::twmax (C++ member), 531, 1049, 1721, 1818  
 sh::ui::qt::QtToolBarButton::tworig1 (C++ member), 531, 1048, 1720, 1817  
 sh::ui::qt::QtToolBarButton::tworig2 (C++ member), 531, 1048, 1720, 1817  
 sh::ui::qt::QtToolBarButton::tworigmax

(C++ member), 531, 1049, 1721, 1818  
sh::ui::qt::QtToolBarButton::unsetClickAction (C++ function), 530, 1048, 1720, 1817  
sh::ui::qt::QtToolBarButton::xi (C++ member), 532, 1049, 1721, 1818  
sh::ui::qt::QtToolBarButton::xt (C++ member), 531, 1049, 1721, 1818  
sh::ui::qt::QtToolBarButton::yi (C++ member), 532, 1049, 1721, 1818  
sh::ui::qt::QtToolBarButton::yt1 (C++ member), 532, 1049, 1721, 1818  
sh::ui::qt::QtToolBarButton::yt2 (C++ member), 532, 1049, 1721, 1818  
sh::ui::qt::QtToolBarButtonHandler (C++ class), 532, 1049, 1721, 1818  
sh::ui::qt::QtToolBarButtonHandler::\_applyPropertiesToButtonDialog (C++ function), 533, 1050, 1722, 1819  
sh::ui::qt::QtToolBarButtonHandler::\_updateSubmenu (C++ function), 532, 1050, 1722, 1819  
sh::ui::qt::QtToolBarButtonHandler::~~QtToolBarButtonHandlerDialog (C++ function), 532, 1050, 1722, 1819  
sh::ui::qt::QtToolBarButtonHandler::actish::ui::qt::QtUIStyle (C++ class), 534, 1051, 1723, 1820  
sh::ui::qt::QtToolBarButtonHandler::buttonsh::ui::qt::QtUIStyle::\_mainWindow (C++ member), 535, 1052, 1724, 1821  
sh::ui::qt::QtToolBarButtonHandler::menush::ui::qt::QtUIStyle::backgroundColor (C++ function), 534, 1052, 1724, 1821  
sh::ui::qt::QtToolBarButtonHandler::preventDefaultActionsh::ui::qt::QtUIStyle::createSheet (C++ function), 535, 1052, 1724, 1821  
sh::ui::qt::QtToolBarButtonHandler::QtToolBarButtonHandlerDialogsh::ui::qt::QtUIStyle::fontSizeInPt (C++ function), 535, 1052, 1724, 1821  
sh::ui::qt::QtTuningDialog (C++ class), 533, 1050, 1722, 1819  
sh::ui::qt::QtTuningDialog::\_changedlglabelsh::ui::qt::QtUIStyle::foregroundColorLighter1 (C++ function), 535, 1052, 1724, 1821  
sh::ui::qt::QtTuningDialog::~~QtTuningDialogsh::ui::qt::QtUIStyle::foregroundColorLighter2 (C++ function), 535, 1052, 1724, 1821  
sh::ui::qt::QtTuningDialog::BoxWidgetByDesktopsh::ui::qt::QtUIStyle::inactiveBackgroundColor (C++ function), 534, 1052, 1724, 1821  
sh::ui::qt::QtTuningDialog::boxWidgetsByDesktopsh::ui::qt::QtUIStyle::linkColor (C++ function), 534, 1052, 1724, 1821  
sh::ui::qt::QtTuningDialog::close (C++ function), 533, 1050, 1722, 1819  
sh::ui::qt::QtTuningDialog::dialogIdsh::ui::qt::QtUIStyle::QtUIStyle (C++ function), 534, 1052, 1724, 1821  
sh::ui::qt::QtTuningDialog::do\_change (C++ function), 534, 1051, 1723, 1820  
sh::ui::qt::QtTuningDialog::filterVisibilitysh::ui::qt::QtUIStyle::Sheet::\_sheet (C++ member), 536, 537, 1053, 1725, 1822  
sh::ui::qt::QtTuningDialog::isInitedsh::ui::qt::QtUIStyle::Sheet::\_style (C++ member), 536, 537, 1053, 1725, 1822  
sh::ui::qt::QtTuningDialog::managersh::ui::qt::QtUIStyle::Sheet::applyTo (C++ function), 535, 536, 1053, 1725, 1822  
sh::ui::qt::QtTuningDialog::on\_btnCancel\_clickedsh::ui::qt::QtUIStyle::Sheet::background

(C++ function), 535, 536, 1052, 1724, 1821  
 sh::ui::qt::QtUIStyle::Sheet::backgroundDarkenedSearchPanelButton::SearchPanelButton  
 (C++ function), 535, 536, 1052, 1724, 1821 (C++ function), 480, 1004, 1676  
 sh::ui::qt::QtUIStyle::Sheet::backgroundHalfHighlightedSearchPanelButton::setButtonText  
 (C++ function), 535, 536, 1052, 1724, 1821 (C++ function), 480, 1004, 1676  
 sh::ui::qt::QtUIStyle::Sheet::backgroundHighlightedSearchPanelButton::setMenuSelection  
 (C++ function), 535, 536, 1052, 1724, 1821 (C++ function), 480, 1004, 1676  
 sh::ui::qt::QtUIStyle::Sheet::backgroundLikeTitlebarSearchPanelConfiguration (C++  
 (C++ function), 535, 536, 1053, 1725, 1822 class), 481, 1004, 1676  
 sh::ui::qt::QtUIStyle::Sheet::bold (C++ sh::ui::SearchPanelConfiguration::~~SearchPanelConf  
 function), 535, 536, 1053, 1725, 1822 (C++ function), 481, 1005, 1676  
 sh::ui::qt::QtUIStyle::Sheet::customcss sh::ui::SearchPanelConfiguration::addActionButton  
 (C++ function), 535, 536, 1052, 1724, 1821 (C++ function), 481, 1005, 1676  
 sh::ui::qt::QtUIStyle::Sheet::fontSize\_byFactorSearchPanelConfiguration::addDateTimeEditor  
 (C++ function), 535, 536, 1052, 1724, 1821 (C++ function), 481, 1005, 1676  
 sh::ui::qt::QtUIStyle::Sheet::fontSize\_header1SearchPanelConfiguration::addLabel  
 (C++ function), 535, 536, 1052, 1724, 1821 (C++ function), 481, 1005, 1676  
 sh::ui::qt::QtUIStyle::Sheet::fontSize\_small1SearchPanelConfiguration::addMenuButton  
 (C++ function), 535, 536, 1052, 1724, 1821 (C++ function), 481, 1005, 1676  
 sh::ui::qt::QtUIStyle::Sheet::Sheet sh::ui::SearchPanelConfiguration::addSpacer  
 (C++ function), 535, 536, 1053, 1725, 1822 (C++ function), 481, 1005, 1676  
 sh::ui::qt::QtUIStyle::Sheet::sheet sh::ui::SearchPanelConfiguration::addTextEditor  
 (C++ function), 535, 536, 1052, 1724, 1821 (C++ function), 481, 1005, 1676  
 sh::ui::qt::QtUIStyle::Sheet::textcolor sh::ui::SearchPanelConfiguration::getEditorAt  
 (C++ function), 535, 536, 1052, 1724, 1821 (C++ function), 481, 1005, 1676  
 sh::ui::qt::QtUIStyle::Sheet::textcolor\_header1SearchPanelConfiguration::myqobject  
 (C++ function), 535, 536, 1052, 1724, 1821 (C++ member), 481, 1005, 1677  
 sh::ui::qt::QtUIStyle::Sheet::textcolor\_header2SearchPanelConfiguration::onDestroyed  
 (C++ function), 535, 536, 1052, 1724, 1821 (C++ function), 481, 1005, 1677  
 sh::ui::qt::QtUIStyle::titlebarColor sh::ui::SearchPanelConfiguration::SearchPanelConfiq  
 (C++ function), 534, 1052, 1724, 1821 (C++ function), 481, 1005, 1676  
 sh::ui::qt::QtUIStyle::windowColor (C++ sh::ui::SearchPanelDateTimeEditor (C++  
 function), 534, 1052, 1724, 1821 class), 481, 1005, 1677  
 sh::ui::qt::QtUIStyle::windowColorDarkened sh::ui::SearchPanelDateTimeEditor::datetime  
 (C++ function), 534, 1052, 1724, 1821 (C++ function), 482, 1005, 1677  
 sh::ui::qt::QtUIStyle::windowColorHalfHighlightedSearchPanelDateTimeEditor::isWidgetEnabled  
 (C++ function), 534, 1052, 1724, 1821 (C++ function), 482, 1005, 1677  
 sh::ui::qt::QtUIStyle::windowColorHighlighted sh::ui::SearchPanelDateTimeEditor::SearchPanelDate  
 (C++ function), 534, 1052, 1724, 1821 (C++ function), 482, 1005, 1677  
 sh::ui::qt::QtUIStyle::windowColorLikeTitlebar sh::ui::SearchPanelDateTimeEditor::setDatetime  
 (C++ function), 534, 1052, 1724, 1821 (C++ function), 482, 1005, 1677  
 sh::ui::SearchPanelAbstractEditor (C++ sh::ui::SearchPanelDateTimeEditor::setWidgetEnabled  
 class), 480, 1004, 1675 (C++ function), 482, 1005, 1677  
 sh::ui::SearchPanelAbstractEditor::~~SearchPanelAbstractEditor (C++  
 (C++ function), 480, 1004, 1676 class), 482, 1005, 1677  
 sh::ui::SearchPanelAbstractEditor::isWidgetEnabledSearchPanelLabelEditor::focusProxyEditor  
 (C++ function), 480, 1004, 1676 (C++ function), 482, 1006, 1677  
 sh::ui::SearchPanelAbstractEditor::SearchPanelAbstractEditorLabelEditor::isWidgetEnabled  
 (C++ function), 480, 1004, 1676 (C++ function), 482, 1006, 1677  
 sh::ui::SearchPanelAbstractEditor::setWidgetEnabledSearchPanelLabelEditor::SearchPanelLabelEd  
 (C++ function), 480, 1004, 1676 (C++ function), 482, 1006, 1677  
 sh::ui::SearchPanelButton (C++ class), 480, sh::ui::SearchPanelLabelEditor::setFocusProxyEditor  
 1004, 1676 (C++ function), 482, 1006, 1677  
 sh::ui::SearchPanelButton::~~SearchPanelButton sh::ui::SearchPanelLabelEditor::setTextContent

(C++ function), 482, 1006, 1677  
sh::ui::SearchPanelLabelEditor::setEnabledStoreProfileDialog::SettingEntry::\_setting  
(C++ function), 482, 1006, 1677 (C++ member), 485, 1008, 1680  
sh::ui::SearchPanelLabelEditor::textContentsh::ui::StoreProfileDialog::SettingEntry::fileview  
(C++ function), 482, 1006, 1677 (C++ function), 485, 1008, 1680  
sh::ui::SearchPanelSpacerEditor (C++ sh::ui::StoreProfileDialog::SettingEntry::isForFile  
class), 482, 1006, 1678 (C++ function), 485, 1008, 1680  
sh::ui::SearchPanelSpacerEditor::isEnabledStoreProfileDialog::SettingEntry::setting  
(C++ function), 483, 1006, 1678 (C++ function), 485, 1008, 1680  
sh::ui::SearchPanelSpacerEditor::SearchPanelSpacerEditorProfileDialog::SettingEntry::SettingEn  
(C++ function), 483, 1006, 1678 (C++ function), 485, 1008, 1680  
sh::ui::SearchPanelSpacerEditor::setEnabledStoreProfileDialog::StoreProfileDialog  
(C++ function), 483, 1006, 1678 (C++ function), 484, 1007, 1678  
sh::ui::SearchPanelTextEditor (C++ class), sh::ui::StoreProfileDialog::waitClosed  
483, 1006, 1678 (C++ function), 484, 1007, 1679  
sh::ui::SearchPanelTextEditor::isEnabledStoreProfileDialog::wasAccepted  
(C++ function), 483, 1006, 1678 (C++ function), 484, 1007, 1679  
sh::ui::SearchPanelTextEditor::placeholderDescriptionStoreProfileDialog::wasClosed  
(C++ function), 483, 1006, 1678 (C++ function), 484, 1007, 1679  
sh::ui::SearchPanelTextEditor::SearchPanelTextEditorStoreProfileDialog::withSubDirectories  
(C++ function), 483, 1006, 1678 (C++ function), 484, 1007, 1679  
sh::ui::SearchPanelTextEditor::setPlaceholderDescriptionTabViewStruct (C++ type), 978, 1650  
(C++ function), 483, 1006, 1678 sh::ui::TabViewStructWidgets (C++ type),  
sh::ui::SearchPanelTextEditor::setTextContent 978, 1650  
(C++ function), 483, 1006, 1678 sh::ui::TuningDialog (C++ class), 486, 1008,  
sh::ui::SearchPanelTextEditor::setEnabled680  
(C++ function), 483, 1007, 1678 sh::ui::TuningDialog::close (C++ function),  
sh::ui::SearchPanelTextEditor::textContent 486, 1009, 1680  
(C++ function), 483, 1006, 1678 sh::ui::TuningDialog::dialogId (C++ func-  
sh::ui::StoreProfileDialog (C++ class), 483, tion), 486, 1008, 1680  
1007, 1678 sh::ui::TuningDialog::isInitd (C++ func-  
sh::ui::StoreProfileDialog::\_eurl (C++ tion), 486, 1008, 1680  
member), 485, 1008, 1679 sh::ui::TuningDialog::manager (C++ func-  
sh::ui::StoreProfileDialog::checkedSettings tion), 486, 1009, 1680  
(C++ function), 484, 1007, 1678 sh::ui::TuningDialog::TuningDialog (C++  
sh::ui::StoreProfileDialog::close (C++ function), 486, 1008, 1680  
function), 484, 1007, 1679 sh::ui::TuningDialog::waitClosed (C++  
sh::ui::StoreProfileDialog::dialogId function), 486, 1008, 1680  
(C++ function), 484, 1007, 1679 sh::ui::TuningDialog::wasAccepted (C++  
sh::ui::StoreProfileDialog::eurl (C++ function), 486, 1008, 1680  
function), 484, 1007, 1678 sh::ui::TuningDialog::wasClosed (C++  
sh::ui::StoreProfileDialog::hasGlobal function), 486, 1009, 1680  
(C++ function), 484, 1007, 1679 sh::ui::web (C++ type), 1058, 1730, 1833  
sh::ui::StoreProfileDialog::inheritsFromsh::ui::web::engines (C++ type), 1106, 1777,  
(C++ function), 484, 1007, 1679 1879  
sh::ui::StoreProfileDialog::isInitd sh::ui::web::WebAboutDialog (C++ class),  
(C++ function), 484, 1007, 1679 542, 1059, 1731, 1833  
sh::ui::StoreProfileDialog::manager sh::ui::web::WebAboutDialog::close (C++  
(C++ function), 484, 1008, 1679 function), 543, 1060, 1732, 1834  
sh::ui::StoreProfileDialog::profileName sh::ui::web::WebAboutDialog::dialogId  
(C++ function), 484, 1007, 1678 (C++ function), 543, 1059, 1731, 1834  
sh::ui::StoreProfileDialog::SettingEntrysh::ui::web::WebAboutDialog::dialogProperty  
(C++ class), 485, 1008, 1679 (C++ function), 543, 1059, 1731, 1833  
sh::ui::StoreProfileDialog::SettingEntrysh::ui::web::WebAboutDialog::dialogResult



Index 2029

(C++ function), 545, 1061, 1733, 1835  
sh::ui::web::WebActionExecutionInfoDialogh:setBackGroundWebActionExecutionInfoPanel::\_width  
(C++ function), 545, 1061, 1733, 1836 (C++ member), 548, 1064, 1736, 1838  
sh::ui::web::WebActionExecutionInfoDialogh:setDetailsWebActionExecutionInfoPanel::~WebAction  
(C++ function), 544, 1060, 1732, 1835 (C++ function), 547, 1063, 1735, 1837  
sh::ui::web::WebActionExecutionInfoDialogh:setForceWebActionExecutionInfoPanel::color  
(C++ function), 545, 1061, 1733, 1836 (C++ function), 547, 1063, 1735, 1837  
sh::ui::web::WebActionExecutionInfoDialogh:setNewWeb::WebActionExecutionInfoPanel::forceForce  
(C++ function), 544, 1060, 1732, 1835 (C++ function), 547, 1063, 1735, 1837  
sh::ui::web::WebActionExecutionInfoDialogh:setLowGlobalWebActionExecutionInfoPanel::id  
(C++ function), 545, 1061, 1733, 1836 (C++ function), 547, 1063, 1735, 1837  
sh::ui::web::WebActionExecutionInfoDialogh:setProgressWebActionExecutionInfoPanel::info  
(C++ function), 544, 1060, 1732, 1835 (C++ function), 547, 1063, 1735, 1837  
sh::ui::web::WebActionExecutionInfoDialogh:simpleWebActionExecutionInfoPanel::isProgress  
(C++ function), 544, 1061, 1733, 1835 (C++ function), 547, 1063, 1735, 1837  
sh::ui::web::WebActionExecutionInfoDialogh:simpleWebActionExecutionInfoPanel::label  
(C++ function), 545, 1061, 1733, 1836 (C++ function), 547, 1063, 1735, 1837  
sh::ui::web::WebActionExecutionInfoDialogh:simpleWebActionExecutionInfoPanel::onClicked  
(C++ function), 545, 1061, 1733, 1836 (C++ function), 547, 1063, 1735, 1837  
sh::ui::web::WebActionExecutionInfoDialogh:unixWebActionExecutionInfoPanel::onDestroy  
(C++ function), 544, 1061, 1733, 1835 (C++ function), 547, 1063, 1735, 1838  
sh::ui::web::WebActionExecutionInfoDialogh:UserFeedbackWebActionExecutionInfoPanel::onVisibili  
(C++ class), 546, 1062, 1734, 1837 (C++ function), 547, 1063, 1735, 1838  
sh::ui::web::WebActionExecutionInfoDialogh:UserFeedbackWebActionExecutionInfoPanel::panelVisi  
(C++ member), 546, 547, 1063, 1734, 1837 (C++ function), 547, 1063, 1735, 1837  
sh::ui::web::WebActionExecutionInfoDialogh:UserFeedbackWebActionExecutionInfoPanel::progress  
(C++ member), 546, 547, 1063, 1734, 1837 (C++ function), 547, 1063, 1735, 1837  
sh::ui::web::WebActionExecutionInfoDialogh:userFeedbackWebActionExecutionInfoPanel::progress  
(C++ function), 545, 1061, 1733, 1836 (C++ function), 547, 1063, 1735, 1837  
sh::ui::web::WebActionExecutionInfoDialogh:WebActionExecutionInfoDialogInfoPanel::setForce  
(C++ function), 544, 1060, 1732, 1835 (C++ function), 547, 1063, 1735, 1837  
sh::ui::web::WebActionExecutionInfoDialogh:web::webWebActionExecutionInfoPanel::setLabel  
(C++ function), 545, 1061, 1733, 1835 (C++ function), 547, 1063, 1735, 1837  
sh::ui::web::WebActionExecutionInfoPanelsh::ui::web::WebActionExecutionInfoPanel::setPanelV  
(C++ class), 547, 1063, 1734, 1837 (C++ function), 547, 1063, 1735, 1837  
sh::ui::web::WebActionExecutionInfoPanelsh::colorweb::WebActionExecutionInfoPanel::setProgre  
(C++ member), 548, 1064, 1736, 1838 (C++ function), 547, 1063, 1735, 1837  
sh::ui::web::WebActionExecutionInfoPanelsh::forceWebWebActionExecutionInfoPanel::setWidth  
(C++ member), 548, 1064, 1736, 1838 (C++ function), 547, 1063, 1735, 1837  
sh::ui::web::WebActionExecutionInfoPanelsh::idi::web::WebActionExecutionInfoPanel::WebAction  
(C++ member), 548, 1064, 1736, 1838 (C++ function), 547, 1063, 1735, 1837  
sh::ui::web::WebActionExecutionInfoPanelsh::info::web::WebActionExecutionInfoPanel::width  
(C++ member), 548, 1064, 1736, 1838 (C++ function), 547, 1063, 1735, 1837  
sh::ui::web::WebActionExecutionInfoPanelsh::isProgressWebActionManager (C++ class),  
(C++ member), 548, 1064, 1736, 1838 548, 1064, 1736, 1838  
sh::ui::web::WebActionExecutionInfoPanelsh::labelweb::WebActionManager::\_infodialogs  
(C++ member), 548, 1064, 1736, 1838 (C++ member), 549, 1065, 1737, 1840  
sh::ui::web::WebActionExecutionInfoPanelsh::panelWebWebActionManager::\_infopanels  
(C++ member), 548, 1064, 1736, 1838 (C++ member), 549, 1065, 1737, 1840  
sh::ui::web::WebActionExecutionInfoPanelsh::progressWebActionManager::\_observeToolbarActio  
(C++ member), 548, 1064, 1736, 1838 (C++ function), 549, 1065, 1737, 1839  
sh::ui::web::WebActionExecutionInfoPanelsh::progressWebWebActionManager::\_permanentToolbarAct  
(C++ member), 548, 1064, 1736, 1838 (C++ member), 549, 1065, 1737, 1840  
sh::ui::web::WebActionExecutionInfoPanelsh::triggerWebWebActionManager::\_triggerActionuiChar

(C++ function), 549, 1065, 1737, 1839  
 sh::ui::web::WebActionManager::\_triggerActionmicweb::WebDetailPanelManager::cmd\_\_link\_click  
 (C++ member), 549, 1065, 1737, 1840  
 sh::ui::web::WebActionManager::\_triggerToolBarRefreshWebDetailPanelManager::executeCommand  
 (C++ member), 549, 1065, 1737, 1840  
 sh::ui::web::WebActionManager::addActionExecutionweb::WebDetailPanelManager::executeCommand  
 (C++ function), 548, 1064, 1736, 1838  
 sh::ui::web::WebActionManager::addInfoPanelsh::ui::web::WebDetailPanelManager::load  
 (C++ function), 548, 1064, 1736, 1838  
 sh::ui::web::WebActionManager::cmd\_\_answeruserfeedbackweb::WebDetailPanelManager::rootCommand  
 (C++ function), 550, 1066, 1738, 1840  
 sh::ui::web::WebActionManager::cmd\_\_execsh::ui::web::WebDetailPanelManager::setEngine  
 (C++ function), 550, 1066, 1738, 1840  
 sh::ui::web::WebActionManager::cmd\_\_get sh::ui::web::WebDetailPanelManager::WebDetailPanelM  
 (C++ function), 550, 1066, 1738, 1840  
 sh::ui::web::WebActionManager::cmd\_\_get\_sh::Mi::web::WebDetailPanelManager::webserver  
 (C++ function), 550, 1066, 1738, 1840  
 sh::ui::web::WebActionManager::cmd\_\_iconsh::ui::web::WebDialog (C++ class), 551, 1067,  
 (C++ function), 550, 1066, 1738, 1840 1739, 1841  
 sh::ui::web::WebActionManager::cmd\_\_panelclickedweb::WebDialog::\_dialogClassName  
 (C++ function), 550, 1066, 1738, 1840  
 sh::ui::web::WebActionManager::executeCommandsh::ui::web::WebDialog::\_properties  
 (C++ function), 549, 1065, 1737, 1839  
 sh::ui::web::WebActionManager::executeCommandsh::ui::web::WebDialog::\_properties  
 (C++ function), 549, 1065, 1737, 1839  
 sh::ui::web::WebActionManager::initialize sh::ui::web::WebDialog::~WebDialog (C++  
 (C++ function), 548, 1064, 1736, 1838 function), 552, 1068, 1740, 1842  
 sh::ui::web::WebActionManager::reActionTextAutelweb::WebDialog::dialogProperty  
 (C++ member), 550, 1066, 1738, 1840  
 sh::ui::web::WebActionManager::refreshToolBar sh::ui::web::WebDialog::dialogResult  
 (C++ function), 548, 1064, 1736, 1838  
 sh::ui::web::WebActionManager::resolveActiononpathweb::WebDialog::isCloseableByUser  
 (C++ function), 549, 1065, 1737, 1839  
 sh::ui::web::WebActionManager::rootCommandsh::ui::web::WebDialog::setCloseableByUser  
 (C++ function), 548, 1064, 1736, 1839  
 sh::ui::web::WebActionManager::setEnginesh::ui::web::WebDialog::setDialogProperty  
 (C++ function), 548, 1064, 1736, 1839  
 sh::ui::web::WebActionManager::WebActionManagerweb::WebDialog::toJson (C++ func-  
 (C++ function), 548, 1064, 1736, 1838 tion), 552, 1068, 1740, 1842  
 sh::ui::web::WebActionManager::webserversh::ui::web::WebDialog::triggerDialogEvent  
 (C++ function), 548, 1064, 1736, 1839  
 sh::ui::web::WebCommandData (C++ type), sh::ui::web::WebDialog::WebDialog (C++  
 1059, 1731, 1833 function), 552, 1068, 1739, 1842  
 sh::ui::web::WebDetailPanelManager (C++ sh::ui::web::WebDialogManager (C++ class),  
 class), 550, 1066, 1738, 1840 553, 1069, 1740, 1843  
 sh::ui::web::WebDetailPanelManager::\_mutex sh::ui::web::WebDialogManager::cmd\_\_change\_dialog\_p  
 (C++ member), 551, 1067, 1739, 1841  
 sh::ui::web::WebDetailPanelManager::\_paneldetailweb::WebDialogManager::cmd\_\_closed\_in\_brows  
 (C++ member), 551, 1067, 1739, 1841  
 sh::ui::web::WebDetailPanelManager::\_websh::ui::web::WebDialogManager::cmd\_\_list\_\_M  
 (C++ member), 551, 1067, 1739, 1841  
 sh::ui::web::WebDetailPanelManager::cmd\_\_getdetailweb::WebDialogManager::createAndShowDialog  
 (C++ function), 551, 1067, 1739, 1841  
 sh::ui::web::WebDetailPanelManager::cmd\_\_getuihumhaiWebDialogManager::executeCommand

(C++ function), 554, 1070, 1741, 1844  
 sh::ui::web::WebDialogManager::executeCommandByObjectWebExceptionDialog::setCloseableByUser  
 (C++ function), 554, 1069, 1741, 1843 (C++ function), 555, 1071, 1742, 1845  
 sh::ui::web::WebDialogManager::getAllDialogIds::web::WebExceptionDialog::setDialogProperty  
 (C++ function), 553, 1069, 1741, 1843 (C++ function), 555, 1070, 1742, 1844  
 sh::ui::web::WebDialogManager::getAllDialogsui::web::WebExceptionDialog::showLoglabelAndDet  
 (C++ function), 553, 1069, 1741, 1843 (C++ function), 555, 1071, 1743, 1845  
 sh::ui::web::WebDialogManager::getDialogByIdui::web::WebExceptionDialog::toJson  
 (C++ function), 553, 1069, 1740, 1843 (C++ function), 555, 1071, 1742, 1845  
 sh::ui::web::WebDialogManager::getDialogForRequestweb::WebExceptionDialog::triggerDialogEvent  
 (C++ function), 554, 1070, 1741, 1844 (C++ function), 555, 1070, 1742, 1844  
 sh::ui::web::WebDialogManager::rootCommand::ui::web::WebExceptionDialog::waitClosed  
 (C++ function), 553, 1069, 1740, 1843 (C++ function), 556, 1071, 1743, 1845  
 sh::ui::web::WebDialogManager::setEnginesh::ui::web::WebExceptionDialog::wasAccepted  
 (C++ function), 553, 1069, 1740, 1843 (C++ function), 556, 1071, 1743, 1845  
 sh::ui::web::WebDialogManager::showDialogh::ui::web::WebExceptionDialog::wasClosed  
 (C++ function), 554, 1070, 1741, 1844 (C++ function), 556, 1072, 1743, 1846  
 sh::ui::web::WebDialogManager::WebDialogManagerweb::WebExceptionDialog::WebExceptionDialog  
 (C++ function), 553, 1069, 1740, 1843 (C++ function), 555, 1070, 1742, 1844  
 sh::ui::web::WebDialogManager::webserversh::ui::web::WebFilePropertyDialog (C++  
 (C++ function), 553, 1069, 1740, 1843 class), 556, 1072, 1744, 1846  
 sh::ui::web::WebExceptionDialog (C++ sh::ui::web::WebFilePropertyDialog::\_childs  
 class), 554, 1070, 1742, 1844 (C++ member), 558, 1074, 1746, 1848  
 sh::ui::web::WebExceptionDialog::answer sh::ui::web::WebFilePropertyDialog::\_cookie  
 (C++ function), 555, 1070, 1742, 1844 (C++ member), 558, 1074, 1746, 1848  
 sh::ui::web::WebExceptionDialog::close sh::ui::web::WebFilePropertyDialog::\_refreshTimer  
 (C++ function), 556, 1071, 1743, 1845 (C++ member), 558, 1074, 1746, 1848  
 sh::ui::web::WebExceptionDialog::cmd\_\_iconsh::ui::web::WebFilePropertyDialog::~WebFileProperty  
 (C++ function), 556, 1072, 1744, 1846 (C++ function), 556, 1072, 1744, 1846  
 sh::ui::web::WebExceptionDialog::detailssh::ui::web::WebFilePropertyDialog::addTabFactory  
 (C++ function), 555, 1071, 1742, 1845 (C++ function), 558, 1073, 1745, 1847  
 sh::ui::web::WebExceptionDialog::dialogIdh::ui::web::WebFilePropertyDialog::close  
 (C++ function), 555, 1071, 1743, 1845 (C++ function), 557, 1073, 1745, 1847  
 sh::ui::web::WebExceptionDialog::dialogPropertyweb::WebFilePropertyDialog::cmd\_\_get\_\_proper  
 (C++ function), 555, 1070, 1742, 1844 (C++ function), 558, 1074, 1746, 1848  
 sh::ui::web::WebExceptionDialog::dialogResultui::web::WebFilePropertyDialog::cmd\_\_refresh\_M  
 (C++ function), 555, 1070, 1742, 1844 (C++ function), 558, 1074, 1746, 1848  
 sh::ui::web::WebExceptionDialog::error1 sh::ui::web::WebFilePropertyDialog::cmd\_\_tabwidget  
 (C++ function), 555, 1071, 1742, 1845 (C++ function), 558, 1074, 1746, 1848  
 sh::ui::web::WebExceptionDialog::error2 sh::ui::web::WebFilePropertyDialog::cmd\_\_trigger\_a  
 (C++ function), 555, 1071, 1742, 1845 (C++ function), 558, 1074, 1746, 1848  
 sh::ui::web::WebExceptionDialog::icon sh::ui::web::WebFilePropertyDialog::createTabViewI  
 (C++ function), 555, 1071, 1743, 1845 (C++ function), 556, 1072, 1744, 1846  
 sh::ui::web::WebExceptionDialog::isCloseableByUserweb::WebFilePropertyDialog::createTabViewTa  
 (C++ function), 555, 1071, 1742, 1845 (C++ function), 556, 1072, 1744, 1846  
 sh::ui::web::WebExceptionDialog::isInitesh::ui::web::WebFilePropertyDialog::createTabViewTe  
 (C++ function), 556, 1071, 1743, 1845 (C++ function), 556, 1072, 1744, 1846  
 sh::ui::web::WebExceptionDialog::managersh::ui::web::WebFilePropertyDialog::dialogId  
 (C++ function), 556, 1072, 1743, 1846 (C++ function), 557, 1073, 1745, 1847  
 sh::ui::web::WebExceptionDialog::mayCancelh::ui::web::WebFilePropertyDialog::dialogProperty  
 (C++ function), 555, 1071, 1743, 1845 (C++ function), 556, 1072, 1744, 1846  
 sh::ui::web::WebExceptionDialog::mayCloseh::ui::web::WebFilePropertyDialog::dialogResult  
 (C++ function), 555, 1071, 1743, 1845 (C++ function), 557, 1072, 1744, 1846  
 sh::ui::web::WebExceptionDialog::mayRetrysh::ui::web::WebFilePropertyDialog::getProperties





(C++ function), 557, 1073, 1745, 1847

sh::ui::web::WebFileView (C++ class), 564, 1076, 1748, 1850

sh::ui::web::WebFileView::\_checkedNodes (C++ member), 565, 1078, 1750, 1852

sh::ui::web::WebFileView::\_hiddenFilesVisible (C++ member), 565, 1078, 1750, 1852

sh::ui::web::WebFileView::\_iconDimensions (C++ member), 565, 1078, 1750, 1852

sh::ui::web::WebFileView::\_id (C++ member), 565, 1078, 1750, 1852

sh::ui::web::WebFileView::\_manager (C++ member), 565, 1078, 1750, 1852

sh::ui::web::WebFileView::\_model (C++ member), 565, 1078, 1750, 1852

sh::ui::web::WebFileView::\_selectedNode (C++ member), 565, 1078, 1750, 1852

sh::ui::web::WebFileView::\_set\_model (C++ function), 565, 1078, 1750, 1852

sh::ui::web::WebFileView::\_sizeformatting (C++ member), 565, 1078, 1750, 1852

sh::ui::web::WebFileView::\_sortColumn (C++ member), 565, 1078, 1750, 1852

sh::ui::web::WebFileView::\_sortOrder (C++ member), 565, 1078, 1750, 1852

sh::ui::web::WebFileView::\_viewmode (C++ member), 565, 1078, 1750, 1852

sh::ui::web::WebFileView::columnDimensions (C++ function), 564, 1076, 1748, 1850

sh::ui::web::WebFileView::currentNodeChanged (C++ function), 565, 1078, 1750, 1852

sh::ui::web::WebFileView::filemodel (C++ function), 565, 1077, 1749, 1851

sh::ui::web::WebFileView::getAllVisibleNodes (C++ function), 564, 1077, 1749, 1851

sh::ui::web::WebFileView::getSizeFormattingMode (C++ function), 564, 1077, 1749, 1851

sh::ui::web::WebFileView::gotoDir (C++ function), 564, 1077, 1749, 1851

sh::ui::web::WebFileView::hiddenFilesVisible (C++ function), 564, 1077, 1749, 1851

sh::ui::web::WebFileView::historyTrackers (C++ function), 564, 1076, 1748, 1850

sh::ui::web::WebFileView::iconDimension (C++ function), 564, 1077, 1749, 1851

sh::ui::web::WebFileView::id (C++ function), 565, 1077, 1749, 1851

sh::ui::web::WebFileView::index (C++ function), 564, 1077, 1749, 1851

sh::ui::web::WebFileView::model (C++ function), 565, 1077, 1749, 1851

sh::ui::web::WebFileView::modelChanged (C++ function), 565, 1078, 1750, 1852

sh::ui::web::WebFileView::node (C++ function), 565, 1077, 1749, 1851

sh::ui::web::WebFileView::reload (C++ function), 565, 1077, 1749, 1851

sh::ui::web::WebFileView::selectedNodes (C++ function), 564, 1077, 1749, 1851

sh::ui::web::WebFileView::selectionChanged (C++ function), 565, 1078, 1750, 1852

sh::ui::web::WebFileView::setCheckedNodes (C++ function), 565, 1078, 1750, 1852

sh::ui::web::WebFileView::setFilemodel (C++ function), 565, 1077, 1749, 1851

sh::ui::web::WebFileView::setHiddenFilesVisible (C++ function), 564, 1077, 1749, 1851

sh::ui::web::WebFileView::setIconDimension (C++ function), 564, 1077, 1749, 1851

sh::ui::web::WebFileView::setSelectedNode (C++ function), 565, 1078, 1750, 1852

sh::ui::web::WebFileView::setSelection (C++ function), 564, 1077, 1749, 1851

sh::ui::web::WebFileView::setSizeFormattingMode (C++ function), 564, 1077, 1749, 1851

sh::ui::web::WebFileView::setSort (C++ function), 564, 1077, 1749, 1851

sh::ui::web::WebFileView::setViewmode (C++ function), 564, 1076, 1748, 1850

sh::ui::web::WebFileView::sortColumn (C++ function), 564, 1077, 1749, 1851

sh::ui::web::WebFileView::sortOrder (C++ function), 564, 1077, 1749, 1851

sh::ui::web::WebFileView::viewmode (C++ function), 564, 1076, 1748, 1850

sh::ui::web::WebFileView::viewOptionChanged (C++ function), 565, 1078, 1750, 1852

sh::ui::web::WebFileView::WebFileView (C++ function), 564, 1076, 1748, 1850

sh::ui::web::WebFileViewManager (C++ class), 566, 1078, 1750, 1852

sh::ui::web::WebFileViewManager::\_fileviews (C++ member), 567, 1080, 1752, 1854

sh::ui::web::WebFileViewManager::\_icurrentfileview (C++ member), 567, 1080, 1752, 1854

sh::ui::web::WebFileViewManager::activeSelectionChanged (C++ function), 566, 1079, 1751, 1853

sh::ui::web::WebFileViewManager::addFileView (C++ function), 566, 1079, 1751, 1853

sh::ui::web::WebFileViewManager::cmd\_\_checkednodes (C++ function), 567, 1080, 1752, 1854

sh::ui::web::WebFileViewManager::cmd\_\_focus\_current (C++ function), 567, 1080, 1752, 1854

sh::ui::web::WebFileViewManager::cmd\_\_get\_\_M (C++ function), 567, 1080, 1752, 1854

sh::ui::web::WebFileViewManager::cmd\_\_list\_view\_items (C++ function), 567, 1080, 1752, 1854

sh::ui::web::WebFileViewManager::cmd\_\_node\_activate (C++ function), 567, 1080, 1752, 1854

(C++ function), 567, 1080, 1752, 1854  
 sh::ui::web::WebFileViewManager::cmd\_selection\_changed (C++ function), 567, 1080, 1752, 1854  
 sh::ui::web::WebFileViewManager::currentFileView (C++ function), 566, 1079, 1751, 1853  
 sh::ui::web::WebFileViewManager::currentFileViewChanged (C++ function), 566, 1079, 1751, 1853  
 sh::ui::web::WebFileViewManager::executeCommand (C++ function), 566, 1079, 1751, 1853  
 sh::ui::web::WebFileViewManager::executeCommandById (C++ function), 567, 1079, 1751, 1853  
 sh::ui::web::WebFileViewManager::fileViewCount (C++ function), 566, 1079, 1751, 1853  
 sh::ui::web::WebFileViewManager::fileViewCountChanged (C++ function), 566, 1079, 1751, 1853  
 sh::ui::web::WebFileViewManager::fileViewOptionsChanged (C++ function), 566, 1079, 1751, 1853  
 sh::ui::web::WebFileViewManager::getFileView (C++ function), 566, 1079, 1751, 1853  
 sh::ui::web::WebFileViewManager::getFileViewIndex (C++ function), 566, 1079, 1751, 1853  
 sh::ui::web::WebFileViewManager::removeFileView (C++ function), 566, 1079, 1751, 1853  
 sh::ui::web::WebFileViewManager::rootCommand (C++ function), 566, 1079, 1751, 1853  
 sh::ui::web::WebFileViewManager::setCurrentFileViewByWebMain (C++ function), 566, 1079, 1751, 1853  
 sh::ui::web::WebFileViewManager::setEngine (C++ function), 566, 1079, 1751, 1853  
 sh::ui::web::WebFileViewManager::webserver (C++ function), 566, 1079, 1751, 1853  
 sh::ui::web::WebI18n (C++ class), 567, 1080, 1752, 1854  
 sh::ui::web::WebI18n::get (C++ function), 567, 1080, 1752, 1854  
 sh::ui::web::WebIconTextBannerItem (C++ type), 1059, 1731, 1833  
 sh::ui::web::WebIconTextBannerItemIcon (C++ type), 1059, 1731, 1833  
 sh::ui::web::WebLogViewDialog (C++ class), 568, 1080, 1752, 1854  
 sh::ui::web::WebLogViewDialog::close (C++ function), 569, 1081, 1753, 1855  
 sh::ui::web::WebLogViewDialog::cmd\_get\_message (C++ function), 569, 1081, 1753, 1855  
 sh::ui::web::WebLogViewDialog::dialogId (C++ function), 568, 1081, 1753, 1855  
 sh::ui::web::WebLogViewDialog::dialogProperty (C++ function), 568, 1080, 1752, 1854  
 sh::ui::web::WebLogViewDialog::dialogResult (C++ function), 568, 1080, 1752, 1854  
 sh::ui::web::WebLogViewDialog::headtext (C++ function), 568, 1081, 1753, 1855  
 sh::ui::web::WebLogViewDialog::isCloseableByUser (C++ function), 568, 1081, 1753, 1854  
 sh::ui::web::WebLogViewDialog::isInit (C++ function), 568, 1081, 1753, 1855  
 sh::ui::web::WebLogViewDialog::manager (C++ function), 569, 1081, 1753, 1855  
 sh::ui::web::WebLogViewDialog::minimumSeverity (C++ function), 568, 1081, 1753, 1855  
 sh::ui::web::WebLogViewDialog::setCloseableByUser (C++ function), 568, 1080, 1752, 1854  
 sh::ui::web::WebLogViewDialog::setDialogProperty (C++ function), 568, 1080, 1752, 1854  
 sh::ui::web::WebLogViewDialog::subheadtext (C++ function), 568, 1081, 1753, 1855  
 sh::ui::web::WebLogViewDialog::toJson (C++ function), 568, 1081, 1753, 1855  
 sh::ui::web::WebLogViewDialog::triggerDialogEvent (C++ function), 568, 1080, 1752, 1854  
 sh::ui::web::WebLogViewDialog::waitClosed (C++ function), 569, 1081, 1753, 1855  
 sh::ui::web::WebLogViewDialog::wasAccepted (C++ function), 568, 1081, 1753, 1855  
 sh::ui::web::WebLogViewDialog::wasClosed (C++ function), 569, 1081, 1753, 1855  
 sh::ui::web::WebLogViewDialog::WebLogViewDialog (C++ function), 568, 1080, 1752, 1854  
 sh::ui::web::WebMainWindow (C++ class), 569, 1081, 1753, 1855  
 sh::ui::web::WebMainWindow::\_actionManager (C++ member), 573, 1086, 1758, 1860  
 sh::ui::web::WebMainWindow::\_closeDirectly (C++ function), 572, 573, 1084, 1085, 1756, 1757, 1858, 1859  
 sh::ui::web::WebMainWindow::\_currentDirectory (C++ member), 573, 1086, 1758, 1860  
 sh::ui::web::WebMainWindow::\_currentProfile (C++ member), 573, 1086, 1758, 1860  
 sh::ui::web::WebMainWindow::\_detailPanelManager (C++ member), 573, 1086, 1758, 1860  
 sh::ui::web::WebMainWindow::\_detailsPanelVisible (C++ member), 573, 1086, 1758, 1860  
 sh::ui::web::WebMainWindow::\_dialogManager (C++ member), 573, 1086, 1758, 1860  
 sh::ui::web::WebMainWindow::\_fileViewManager (C++ member), 573, 1086, 1758, 1860  
 sh::ui::web::WebMainWindow::\_initialize (C++ function), 571, 1083, 1755, 1857  
 sh::ui::web::WebMainWindow::\_lastCloseableState (C++ member), 573, 1086, 1758, 1860  
 sh::ui::web::WebMainWindow::\_treeSticky (C++ member), 573, 1086, 1758, 1860  
 sh::ui::web::WebMainWindow::\_webMainWindow (C++ member), 574, 1086, 1758, 1860  
 sh::ui::web::WebMainWindow::\_webserver (C++ member), 573, 1086, 1758, 1860

sh::ui::web::WebMainWindow::~~WebMainWindow (C++ function), 572, 1084, 1756, 1858  
 (C++ function), 569, 1082, 1754, 1856 sh::ui::web::WebMainWindow::isClosing  
 sh::ui::web::WebMainWindow::actionManager (C++ function), 572, 1085, 1757, 1859  
 (C++ function), 573, 1086, 1758, 1860 sh::ui::web::WebMainWindow::isReady  
 sh::ui::web::WebMainWindow::addFileView (C++ function), 573, 1085, 1757, 1859  
 (C++ function), 569, 1082, 1754, 1856 sh::ui::web::WebMainWindow::jumpbarIsTextMode  
 sh::ui::web::WebMainWindow::addInfoPanel (C++ function), 570, 1082, 1754, 1856  
 (C++ function), 570, 572, 1082, 1084, 1754, 1756, 1856, 1858 sh::ui::web::WebMainWindow::jumpbarSetButtonMode  
 (C++ function), 570, 1082, 1754, 1856  
 sh::ui::web::WebMainWindow::closeApp sh::ui::web::WebMainWindow::jumpbarSetTextMode  
 (C++ function), 572, 1084, 1756, 1858 (C++ function), 570, 1082, 1754, 1856  
 sh::ui::web::WebMainWindow::cmd\_filesystem\_get\_info sh::ui::web::WebMainWindow::jumpToEurl  
 (C++ function), 574, 1086, 1758, 1860 (C++ function), 571, 1084, 1756, 1858  
 sh::ui::web::WebMainWindow::cmd\_filesystem\_list\_urls sh::ui::web::WebMainWindow::jumpToNode  
 (C++ function), 574, 1086, 1758, 1860 (C++ function), 570, 1082, 1754, 1856  
 sh::ui::web::WebMainWindow::cmd\_log\_as\_text sh::ui::web::WebMainWindow::mainWindow  
 (C++ function), 574, 1086, 1758, 1860 (C++ function), 572, 1085, 1757, 1859  
 sh::ui::web::WebMainWindow::cmd\_log\_log\_message sh::ui::web::WebMainWindow::onCurrentDirectoryChange  
 (C++ function), 574, 1086, 1758, 1860 (C++ function), 571, 1084, 1756, 1858  
 sh::ui::web::WebMainWindow::cmd\_mainwindow\_set\_curl sh::ui::web::WebMainWindow::onCurrentFileViewChange  
 (C++ function), 574, 1086, 1758, 1860 (C++ function), 571, 1084, 1756, 1858  
 sh::ui::web::WebMainWindow::cmd\_mainwindow\_show\_profile sh::ui::web::WebMainWindow::onCurrentProfileChange  
 (C++ function), 574, 1086, 1758, 1860 (C++ function), 571, 1084, 1756, 1858  
 sh::ui::web::WebMainWindow::cmd\_mainwindow\_show\_webview sh::ui::web::WebMainWindow::onFileViewCountChanged  
 (C++ function), 574, 1086, 1758, 1860 (C++ function), 571, 1084, 1756, 1858  
 sh::ui::web::WebMainWindow::createActionExecution sh::ui::web::WebMainWindow::onFileViewOptionsChange  
 (C++ function), 571, 1083, 1755, 1857 (C++ function), 571, 1083, 1755, 1857  
 sh::ui::web::WebMainWindow::currentDirectory sh::ui::web::WebMainWindow::onGlobalViewOptionsChange  
 (C++ function), 570, 1082, 1754, 1856 (C++ function), 571, 1084, 1756, 1858  
 sh::ui::web::WebMainWindow::currentFileView sh::ui::web::WebMainWindow::openUrlOnDesktop  
 (C++ function), 569, 1082, 1754, 1856 (C++ function), 572, 1085, 1757, 1859  
 sh::ui::web::WebMainWindow::currentProfile sh::ui::web::WebMainWindow::reloadProfile  
 (C++ function), 572, 1084, 1756, 1858 (C++ function), 572, 1084, 1756, 1858  
 sh::ui::web::WebMainWindow::detailPanelPosition sh::ui::web::WebMainWindow::removeFileView  
 (C++ function), 570, 1083, 1755, 1857 (C++ function), 569, 1082, 1754, 1856  
 sh::ui::web::WebMainWindow::dialogManager sh::ui::web::WebMainWindow::rootCommand  
 (C++ function), 569, 1082, 1754, 1856 (C++ function), 572, 1085, 1757, 1859  
 sh::ui::web::WebMainWindow::executeCommand sh::ui::web::WebMainWindow::runsHeadless  
 (C++ function), 573, 1085, 1757, 1859 (C++ function), 573, 1085, 1757, 1859  
 sh::ui::web::WebMainWindow::fileDetailsPanelVisible sh::ui::web::WebMainWindow::setCloseable  
 (C++ function), 572, 1084, 1756, 1858 (C++ function), 573, 1086, 1758, 1860  
 sh::ui::web::WebMainWindow::fileViewCount sh::ui::web::WebMainWindow::setCurrentEurlByNode  
 (C++ function), 569, 1082, 1754, 1856 (C++ function), 573, 1086, 1758, 1860  
 sh::ui::web::WebMainWindow::fileViewsReleased sh::ui::web::WebMainWindow::setCurrentProfile  
 (C++ function), 571, 1083, 1755, 1857 (C++ function), 572, 1084, 1756, 1858  
 sh::ui::web::WebMainWindow::getFileView sh::ui::web::WebMainWindow::setDetailPanelPosition  
 (C++ function), 569, 1082, 1754, 1856 (C++ function), 570, 1083, 1755, 1857  
 sh::ui::web::WebMainWindow::handleCloseApp sh::ui::web::WebMainWindow::setEngine  
 (C++ function), 571, 1083, 1755, 1857 (C++ function), 572, 1085, 1757, 1859  
 sh::ui::web::WebMainWindow::handleClosed sh::ui::web::WebMainWindow::setFileDetailsPanelVisible  
 (C++ function), 572, 1085, 1757, 1859 (C++ function), 570, 1082, 1754, 1856  
 sh::ui::web::WebMainWindow::initialize sh::ui::web::WebMainWindow::setMainWindow  
 (C++ function), 569, 1082, 1754, 1856 (C++ function), 572, 1085, 1757, 1859  
 sh::ui::web::WebMainWindow::isCloseable sh::ui::web::WebMainWindow::setTitlePattern



(C++ function), 570, 1082, 1754, 1856  
 sh::ui::web::WebMainWindow::setToolBarPosition (C++ function), 570, 1083, 1755, 1857  
 sh::ui::web::WebMainWindow::setTreeSticky (C++ function), 570, 1082, 1754, 1856  
 sh::ui::web::WebMainWindow::setTreeVisible (C++ function), 570, 1082, 1754, 1856  
 sh::ui::web::WebMainWindow::showAboutDialog (C++ function), 570, 1083, 1755, 1857  
 sh::ui::web::WebMainWindow::showErrorPanel (C++ function), 572, 1084, 1756, 1858  
 sh::ui::web::WebMainWindow::showFilePropertyDialog (C++ function), 571, 1083, 1755, 1857  
 sh::ui::web::WebMainWindow::showLogViewDialog (C++ function), 571, 1083, 1755, 1857  
 sh::ui::web::WebMainWindow::showManageBookmarksDialog (C++ function), 571, 1083, 1755, 1857  
 sh::ui::web::WebMainWindow::showManageProfilesDialog (C++ function), 570, 1083, 1755, 1857  
 sh::ui::web::WebMainWindow::showOpenWithDialog (C++ function), 570, 1083, 1755, 1857  
 sh::ui::web::WebMainWindow::showStoreProfileDialog (C++ function), 570, 1083, 1755, 1857  
 sh::ui::web::WebMainWindow::showTuningDialog (C++ function), 571, 1083, 1755, 1857  
 sh::ui::web::WebMainWindow::titlePattern (C++ function), 571, 1084, 1756, 1858  
 sh::ui::web::WebMainWindow::toolbarPosition (C++ function), 570, 1083, 1755, 1857  
 sh::ui::web::WebMainWindow::treeSticky (C++ function), 572, 1084, 1756, 1858  
 sh::ui::web::WebMainWindow::treeVisible (C++ function), 572, 1084, 1756, 1858  
 sh::ui::web::WebMainWindow::tryCreateMainWindow (C++ function), 572, 1085, 1757, 1859  
 sh::ui::web::WebMainWindow::uiMode (C++ function), 573, 1085, 1757, 1859  
 sh::ui::web::WebMainWindow::WebMainWindow (C++ function), 569, 1082, 1754, 1856  
 sh::ui::web::WebMainWindow::webserver (C++ function), 569, 1082, 1754, 1856  
 sh::ui::web::WebManageBookmarksDialog (C++ class), 574, 1087, 1759, 1861  
 sh::ui::web::WebManageBookmarksDialog::close (C++ function), 575, 1088, 1760, 1862  
 sh::ui::web::WebManageBookmarksDialog::cmd::add\_bookmark (C++ function), 575, 1088, 1760, 1862  
 sh::ui::web::WebManageBookmarksDialog::cmd::delete\_bookmark (C++ function), 575, 1088, 1760, 1862  
 sh::ui::web::WebManageBookmarksDialog::cmd::get\_bookmark (C++ function), 575, 1088, 1760, 1862  
 sh::ui::web::WebManageBookmarksDialog::cmd::move\_bookmark (C++ function), 575, 1088, 1760, 1862  
 sh::ui::web::WebManageBookmarksDialog::cmd::move\_bookmark\_id (C++ function), 575, 1088, 1760, 1862  
 sh::ui::web::WebManageBookmarksDialog::cmd::move\_bookmark\_name (C++ function), 575, 1088, 1760, 1862  
 sh::ui::web::WebManageBookmarksDialog::cmd::store (C++ function), 575, 1088, 1760, 1862  
 sh::ui::web::WebManageBookmarksDialog::dialogId (C++ function), 575, 1087, 1759, 1861  
 sh::ui::web::WebManageBookmarksDialog::dialogProperties (C++ function), 574, 1087, 1759, 1861  
 sh::ui::web::WebManageBookmarksDialog::dialogResult (C++ function), 574, 1087, 1759, 1861  
 sh::ui::web::WebManageBookmarksDialog::isCloseable (C++ function), 575, 1087, 1759, 1861  
 sh::ui::web::WebManageBookmarksDialog::isInitiated (C++ function), 575, 1087, 1759, 1861  
 sh::ui::web::WebManageBookmarksDialog::manager (C++ function), 575, 1088, 1760, 1862  
 sh::ui::web::WebManageBookmarksDialog::setCloseable (C++ function), 574, 1087, 1759, 1861  
 sh::ui::web::WebManageBookmarksDialog::setDialogProperties (C++ function), 574, 1087, 1759, 1861  
 sh::ui::web::WebManageBookmarksDialog::toJson (C++ function), 575, 1087, 1759, 1861  
 sh::ui::web::WebManageBookmarksDialog::triggerDialog (C++ function), 574, 1087, 1759, 1861  
 sh::ui::web::WebManageBookmarksDialog::waitClosed (C++ function), 575, 1088, 1760, 1862  
 sh::ui::web::WebManageBookmarksDialog::wasAccepted (C++ function), 575, 1087, 1759, 1861  
 sh::ui::web::WebManageBookmarksDialog::wasClosed (C++ function), 575, 1088, 1760, 1862  
 sh::ui::web::WebManageBookmarksDialog::WebManageBookmarksDialog (C++ class), 574, 1087, 1759, 1861  
 sh::ui::web::WebManageProfilesDialog (C++ class), 576, 1088, 1760, 1862  
 sh::ui::web::WebManageProfilesDialog::close (C++ function), 576, 1089, 1761, 1863  
 sh::ui::web::WebManageProfilesDialog::cmd::get\_M (C++ function), 577, 1089, 1761, 1863  
 sh::ui::web::WebManageProfilesDialog::cmd::icon (C++ function), 577, 1089, 1761, 1863  
 sh::ui::web::WebManageProfilesDialog::cmd::remove\_r (C++ function), 577, 1089, 1761, 1863  
 sh::ui::web::WebManageProfilesDialog::cmd::remove\_p (C++ function), 577, 1089, 1761, 1863  
 sh::ui::web::WebManageProfilesDialog::dialogId (C++ function), 576, 1089, 1761, 1863  
 sh::ui::web::WebManageProfilesDialog::dialogProperties (C++ function), 576, 1088, 1760, 1862  
 sh::ui::web::WebManageProfilesDialog::dialogResult (C++ function), 576, 1088, 1760, 1862  
 sh::ui::web::WebManageProfilesDialog::isCloseableBy (C++ function), 576, 1089, 1761, 1863  
 sh::ui::web::WebManageProfilesDialog::isInitiated (C++ function), 576, 1089, 1761, 1863

(C++ function), 576, 1089, 1761, 1863  
 sh::ui::web::WebManageProfilesDialog::manager (C++ function), 577, 1089, 1761, 1863  
 sh::ui::web::WebManageProfilesDialog::setCloseableByUser (C++ function), 576, 1088, 1760, 1862  
 sh::ui::web::WebManageProfilesDialog::setDialogPropertyWebOpenWithDialog::isInitied (C++ function), 576, 1088, 1760, 1862  
 sh::ui::web::WebManageProfilesDialog::toJsonsh::ui::web::WebOpenWithDialog::manager (C++ function), 576, 1089, 1761, 1863  
 sh::ui::web::WebManageProfilesDialog::triggerDialogEventWebOpenWithDialog::node (C++ function), 576, 1088, 1760, 1862  
 sh::ui::web::WebManageProfilesDialog::wasClosedweb::WebOpenWithDialog::openMethods (C++ function), 576, 1089, 1761, 1863  
 sh::ui::web::WebManageProfilesDialog::wasAcceptedweb::WebOpenWithDialog::rememberForDirectory (C++ function), 576, 1089, 1761, 1863  
 sh::ui::web::WebManageProfilesDialog::wasClosedweb::WebOpenWithDialog::rememberForFile (C++ function), 577, 1089, 1761, 1863  
 sh::ui::web::WebManageProfilesDialog::WebManageProfileWebOpenWithDialog::rememberForMimetype (C++ function), 576, 1088, 1760, 1862  
 sh::ui::web::WebModule (C++ class), 577, 1089, 1761, 1863  
 sh::ui::web::WebModule::\_webserver (C++ member), 578, 1090, 1762, 1864  
 sh::ui::web::WebModule::~WebModule (C++ function), 577, 1090, 1762, 1864  
 sh::ui::web::WebModule::executeCommand (C++ function), 577, 1090, 1762, 1864  
 sh::ui::web::WebModule::executeCommand\_byQObjectRefWebOpenWithDialog::waitClosed (C++ function), 578, 1090, 1762, 1864  
 sh::ui::web::WebModule::rootCommand (C++ function), 577, 1090, 1762, 1864  
 sh::ui::web::WebModule::setEngine (C++ function), 577, 1090, 1762, 1864  
 sh::ui::web::WebModule::WebModule (C++ function), 577, 1090, 1762, 1864  
 sh::ui::web::WebModule::webserver (C++ function), 577, 1090, 1762, 1864  
 sh::ui::web::WebOpenWithDialog (C++ class), 578, 1090, 1762, 1864  
 sh::ui::web::WebOpenWithDialog::chosenMethodsh::ui::web::WebServerEngine::\_configValues (C++ member), 581, 1094, 1766, 1867  
 sh::ui::web::WebOpenWithDialog::close (C++ function), 579, 1092, 1764, 1866  
 sh::ui::web::WebOpenWithDialog::cmd\_\_check\_anotherwebanWebServerEngine::\_currentTimestamp (C++ member), 581, 1094, 1766, 1868  
 sh::ui::web::WebOpenWithDialog::cmd\_\_check\_customwebWebServerEngine::\_dispatcherurl (C++ member), 581, 1094, 1766, 1867  
 sh::ui::web::WebOpenWithDialog::cmd\_\_openshmethodwebonWebServerEngine::\_externalurl (C++ member), 581, 1094, 1766, 1867  
 sh::ui::web::WebOpenWithDialog::cmd\_\_openshmethodwebMWebServerEngine::\_jok (C++ member), 581, 1094, 1766, 1867  
 sh::ui::web::WebOpenWithDialog::dialogIdsh::ui::web::WebServerEngine::\_mainmodule (C++ member), 581, 1094, 1766, 1867  
 sh::ui::web::WebOpenWithDialog::dialogPropertysh::ui::web::WebServerEngine::\_maxport (C++ member), 581, 1094, 1766, 1867



(C++ function), 584, 586, 1096, 1768, 1870	(C++ function), 585, 587, 1097, 1769, 1871
sh::ui::web::WebServerEngineDispatcher::ShardorThebadWebSepwerKagzneDispatcher::WorkerProce	sh::ui::web::WebServerEngineDispatcher::ShardorThebadWebSepwerKagzneDispatcher::WorkerProce
(C++ function), 584, 586, 1096, 1768, 1870	(C++ function), 584, 587, 1097, 1769, 1871
sh::ui::web::WebServerEngineDispatcher::shotCommonweb::WebServerEngineDispatcher::WorkerProce	sh::ui::web::WebServerEngineDispatcher::shotCommonweb::WebServerEngineDispatcher::WorkerProce
(C++ function), 583, 1095, 1767, 1869	(C++ function), 585, 587, 1097, 1769, 1871
sh::ui::web::WebServerEngineDispatcher::shEngineweb::WebServerEngineDispatcher::WorkerProce	sh::ui::web::WebServerEngineDispatcher::shEngineweb::WebServerEngineDispatcher::WorkerProce
(C++ function), 582, 1094, 1766, 1868	(C++ function), 584, 587, 1097, 1769, 1871
sh::ui::web::WebServerEngineDispatcher::spawnNewWebkeWebServerEngineDispatcher::WorkerProce	sh::ui::web::WebServerEngineDispatcher::spawnNewWebkeWebServerEngineDispatcher::WorkerProce
(C++ function), 583, 1095, 1767, 1869	(C++ function), 584, 587, 1097, 1769, 1871
sh::ui::web::WebServerEngineDispatcher::sharti::web::WebServerEngineDispatcher::WorkerProce	sh::ui::web::WebServerEngineDispatcher::sharti::web::WebServerEngineDispatcher::WorkerProce
(C++ function), 583, 1096, 1768, 1870	(C++ function), 585, 588, 1098, 1770, 1872
sh::ui::web::WebServerEngineDispatcher::shopui::web::WebServerEngineDispatcher::WorkerProce	sh::ui::web::WebServerEngineDispatcher::shopui::web::WebServerEngineDispatcher::WorkerProce
(C++ function), 582, 1094, 1766, 1868	(C++ function), 585, 588, 1098, 1770, 1871
sh::ui::web::WebServerEngineDispatcher::webseiweweb::WebServerEngineDispatcher::WorkerProce	sh::ui::web::WebServerEngineDispatcher::webseiweweb::WebServerEngineDispatcher::WorkerProce
(C++ function), 582, 1094, 1766, 1868	(C++ function), 584, 587, 1097, 1769, 1871
sh::ui::web::WebServerEngineDispatcher::WebSeiwewEhgineWdSepatcerEngineDispatcher::WorkerProce	sh::ui::web::WebServerEngineDispatcher::WebSeiwewEhgineWdSepatcerEngineDispatcher::WorkerProce
(C++ function), 582, 1094, 1766, 1868	(C++ function), 585, 588, 1098, 1770, 1871
sh::ui::web::WebServerEngineDispatcher::WorkaiPwebsWebServerEngineDispatcher::WorkerProce	sh::ui::web::WebServerEngineDispatcher::WorkaiPwebsWebServerEngineDispatcher::WorkerProce
(C++ class), 584, 587, 1097, 1769, 1870	(C++ function), 584, 587, 1097, 1769, 1871
sh::ui::web::WebServerEngineDispatcher::WorkaiPwebsWebSaiwatEngineDispatcher::wtokenFromM	sh::ui::web::WebServerEngineDispatcher::WorkaiPwebsWebSaiwatEngineDispatcher::wtokenFromM
(C++ member), 586, 588, 1098, 1770, 1872	(C++ function), 583, 1095, 1767, 1869
sh::ui::web::WebServerEngineDispatcher::WorkaiPwebsWebSadventEngneMainModule	sh::ui::web::WebServerEngineDispatcher::WorkaiPwebsWebSadventEngneMainModule
(C++ member), 586, 588, 1098, 1770, 1872	(C++ class), 589, 1098, 1770, 1872
sh::ui::web::WebServerEngineDispatcher::WorkaiPwebsWebSaspatEhgineMainModule::_eventEntr	sh::ui::web::WebServerEngineDispatcher::WorkaiPwebsWebSaspatEhgineMainModule::_eventEntr
(C++ member), 586, 588, 1098, 1770, 1872	(C++ member), 590, 1100, 1771, 1873
sh::ui::web::WebServerEngineDispatcher::WorkaiPwebsWebSestBrEvgenHMainMedtle::_eventEntr	sh::ui::web::WebServerEngineDispatcher::WorkaiPwebsWebSestBrEvgenHMainMedtle::_eventEntr
(C++ member), 586, 588, 1098, 1770, 1872	(C++ member), 590, 1100, 1771, 1873
sh::ui::web::WebServerEngineDispatcher::WorkaiPwebsWebSestVrEngneMainModule::_eventEntr	sh::ui::web::WebServerEngineDispatcher::WorkaiPwebsWebSestVrEngneMainModule::_eventEntr
(C++ member), 586, 588, 1098, 1770, 1872	(C++ member), 590, 1100, 1771, 1873
sh::ui::web::WebServerEngineDispatcher::WorkaiPwebsWebSerwerEngineMainModule::_eventEntry	sh::ui::web::WebServerEngineDispatcher::WorkaiPwebsWebSerwerEngineMainModule::_eventEntry
(C++ member), 586, 588, 1098, 1770, 1872	(C++ member), 590, 1100, 1771, 1873
sh::ui::web::WebServerEngineDispatcher::WorkaiPwebsWebSerderEngineMainModule::_lastReques	sh::ui::web::WebServerEngineDispatcher::WorkaiPwebsWebSerderEngineMainModule::_lastReques
(C++ member), 586, 588, 1098, 1770, 1872	(C++ member), 590, 1100, 1771, 1873
sh::ui::web::WebServerEngineDispatcher::WorkaiPwebsWebSeerwenEdgoseMainMdgeule::_mutex_ever	sh::ui::web::WebServerEngineDispatcher::WorkaiPwebsWebSeerwenEdgoseMainMdgeule::_mutex_ever
(C++ member), 586, 588, 1098, 1770, 1872	(C++ member), 590, 1100, 1771, 1873
sh::ui::web::WebServerEngineDispatcher::WorkaiPwebsWebSePrerEngineMainModule::_mutex_last	sh::ui::web::WebServerEngineDispatcher::WorkaiPwebsWebSePrerEngineMainModule::_mutex_last
(C++ member), 586, 588, 1098, 1770, 1872	(C++ member), 590, 1100, 1771, 1873
sh::ui::web::WebServerEngineDispatcher::WorkaiPwebsWebSeotverEngineMainModule::_restatic	sh::ui::web::WebServerEngineDispatcher::WorkaiPwebsWebSeotverEngineMainModule::_restatic
(C++ member), 586, 588, 1098, 1770, 1872	(C++ member), 590, 1100, 1772, 1874
sh::ui::web::WebServerEngineDispatcher::WorkaiPwebsWebStokenEngineMainModule::_stopWhenIo	sh::ui::web::WebServerEngineDispatcher::WorkaiPwebsWebStokenEngineMainModule::_stopWhenIo
(C++ member), 586, 588, 1098, 1770, 1872	(C++ member), 590, 1100, 1772, 1873
sh::ui::web::WebServerEngineDispatcher::WorkaiPwebsWebSeentHngineMainModule::~~WebServerI	sh::ui::web::WebServerEngineDispatcher::WorkaiPwebsWebSeentHngineMainModule::~~WebServerI
(C++ function), 585, 587, 1097, 1769, 1871	(C++ function), 589, 1099, 1771, 1872
sh::ui::web::WebServerEngineDispatcher::WorkaiPwebsWebSeBrowEngneMainModule::cmd_dispatc	sh::ui::web::WebServerEngineDispatcher::WorkaiPwebsWebSeBrowEngneMainModule::cmd_dispatc
(C++ function), 585, 588, 1098, 1770, 1871	(C++ function), 590, 1100, 1772, 1873
sh::ui::web::WebServerEngineDispatcher::WorkaiPwebsWebSeUverEngneMainModule::cmd_events_	sh::ui::web::WebServerEngineDispatcher::WorkaiPwebsWebSeUverEngneMainModule::cmd_events_
(C++ function), 585, 588, 1098, 1770, 1871	(C++ function), 590, 1100, 1772, 1873
sh::ui::web::WebServerEngineDispatcher::WorkaiPwebsWebServaiEngneMainModule::EventEntry	sh::ui::web::WebServerEngineDispatcher::WorkaiPwebsWebServaiEngneMainModule::EventEntry
(C++ function), 585, 588, 1098, 1770, 1871	(C++ class), 590, 591, 1100, 1772, 1874
sh::ui::web::WebServerEngineDispatcher::WorkaiPwebsWebSeioeiEngneMainModule::EventEntry	sh::ui::web::WebServerEngineDispatcher::WorkaiPwebsWebSeioeiEngneMainModule::EventEntry
(C++ function), 585, 587, 1097, 1769, 1871	(C++ member), 590, 591, 1100, 1772, 1874
sh::ui::web::WebServerEngineDispatcher::WorkaiPwebsWebSerDerEngneMainModule::EventEntry	sh::ui::web::WebServerEngineDispatcher::WorkaiPwebsWebSerDerEngneMainModule::EventEntry
(C++ function), 584, 587, 1097, 1769, 1871	(C++ function), 590, 591, 1100, 1772, 1874
sh::ui::web::WebServerEngineDispatcher::WorkaiPwebsWebSerUserEngneMainModule::EventEntry	sh::ui::web::WebServerEngineDispatcher::WorkaiPwebsWebSerUserEngneMainModule::EventEntry



(C++ member), 590, 591, 1100, 1772, 1874  
 sh::ui::web::WebServerEngineMainModule::EventWeb::WebServerEngineRequest::getCookie  
 (C++ member), 590, 591, 1100, 1772, 1874 (C++ function), 593, 1102, 1773, 1875  
 sh::ui::web::WebServerEngineMainModule::EventWeb::WebServerEngineRequest::headerData  
 (C++ member), 590, 591, 1100, 1772, 1874 (C++ function), 593, 1102, 1773, 1875  
 sh::ui::web::WebServerEngineMainModule::EventWeb::WebServerEngineRequest::HTTP\_BAD\_GATEWAY  
 (C++ class), 591, 592, 1100, 1772, 1874 (C++ member), 593, 1102, 1774, 1876  
 sh::ui::web::WebServerEngineMainModule::EventWeb::WebServerEngineRequest::HTTP\_INTERNAL\_SERVER\_ERROR  
 (C++ function), 591, 592, 1101, 1772, 1874 (C++ member), 593, 1102, 1774, 1876  
 sh::ui::web::WebServerEngineMainModule::EventWeb::WebServerEngineRequest::HTTP\_NOT\_FOUND  
 (C++ member), 591, 592, 1101, 1772, 1874 (C++ member), 593, 1102, 1774, 1876  
 sh::ui::web::WebServerEngineMainModule::executeCommandWebServerEngineRequest::HTTP\_OK  
 (C++ function), 589, 1099, 1771, 1872 (C++ member), 593, 1102, 1774, 1876  
 sh::ui::web::WebServerEngineMainModule::executeCommandWebServerEngineRequest::HTTP\_SEE\_OTHER  
 (C++ function), 589, 1099, 1771, 1873 (C++ member), 593, 1102, 1774, 1876  
 sh::ui::web::WebServerEngineMainModule::lastRequestWebServerEngineRequest::path  
 (C++ function), 590, 1099, 1771, 1873 (C++ function), 592, 1101, 1773, 1875  
 sh::ui::web::WebServerEngineMainModule::shotCommandWebServerEngineRequest::response  
 (C++ function), 589, 1099, 1771, 1872 (C++ function), 593, 1101, 1773, 1875  
 sh::ui::web::WebServerEngineMainModule::setEngineWebServerEngineRequest::responseCode  
 (C++ function), 589, 1099, 1771, 1872 (C++ function), 593, 1101, 1773, 1875  
 sh::ui::web::WebServerEngineMainModule::shopWhenWebServerEngineRequest::responseMimeType  
 (C++ class), 591, 592, 1101, 1772, 1874 (C++ function), 593, 1101, 1773, 1875  
 sh::ui::web::WebServerEngineMainModule::shopWhenWebServerEngineRequest::responseSet  
 (C++ member), 591, 592, 1101, 1773, 1875 (C++ function), 593, 1101, 1773, 1875  
 sh::ui::web::WebServerEngineMainModule::shopWhenWebServerEngineRequest::setCookie  
 (C++ function), 591, 592, 1101, 1773, 1874 (C++ function), 593, 1102, 1773, 1875  
 sh::ui::web::WebServerEngineMainModule::triggerEventWebServerEngineRequest::setResponse  
 (C++ function), 589, 1099, 1771, 1872 (C++ function), 593, 1101, 1102, 1773, 1875  
 sh::ui::web::WebServerEngineMainModule::webserverWebServerEngineRequest::url  
 (C++ function), 589, 1099, 1771, 1873 (C++ function), 592, 1101, 1773, 1875  
 sh::ui::web::WebServerEngineMainModule::WebServerEngineMainModuleWebServerEngineRequest::WebServerEngineMainModule  
 (C++ function), 589, 1099, 1771, 1872 (C++ function), 592, 1101, 1773, 1875  
 sh::ui::web::WebServerEngineRequest sh::ui::web::WebStoreProfileDialog (C++  
 (C++ class), 592, 1101, 1773, 1875 class), 594, 1102, 1774, 1876  
 sh::ui::web::WebServerEngineRequest::\_clientHostWebStoreProfileDialog::checkedSettings  
 (C++ member), 594, 1102, 1774, 1876 (C++ function), 594, 1103, 1774, 1876  
 sh::ui::web::WebServerEngineRequest::\_getDataWebStoreProfileDialog::close  
 (C++ member), 594, 1102, 1774, 1876 (C++ function), 595, 1104, 1775, 1877  
 sh::ui::web::WebServerEngineRequest::\_pathWebStoreProfileDialog::cmd\_get\_\_M  
 (C++ member), 594, 1102, 1774, 1876 (C++ function), 595, 1104, 1776, 1878  
 sh::ui::web::WebServerEngineRequest::\_responseWebStoreProfileDialog::cmd\_store\_mode  
 (C++ member), 594, 1102, 1774, 1876 (C++ function), 595, 1104, 1776, 1878  
 sh::ui::web::WebServerEngineRequest::\_responseCodeWebStoreProfileDialog::dialogId  
 (C++ member), 594, 1102, 1774, 1876 (C++ function), 595, 1103, 1775, 1877  
 sh::ui::web::WebServerEngineRequest::\_responseMimeTypeWebStoreProfileDialog::dialogProperty  
 (C++ member), 594, 1102, 1774, 1876 (C++ function), 594, 1103, 1774, 1876  
 sh::ui::web::WebServerEngineRequest::\_responseSetWebStoreProfileDialog::dialogResult  
 (C++ member), 594, 1102, 1774, 1876 (C++ function), 594, 1103, 1775, 1876  
 sh::ui::web::WebServerEngineRequest::\_urlWebStoreProfileDialog::eurl  
 (C++ member), 594, 1102, 1774, 1876 (C++ function), 595, 1103, 1775, 1877  
 sh::ui::web::WebServerEngineRequest::clientHostWebStoreProfileDialog::hasGlobal  
 (C++ function), 592, 1101, 1773, 1875 (C++ function), 594, 1103, 1774, 1876  
 sh::ui::web::WebServerEngineRequest::dataWebStoreProfileDialog::inheritsFrom

(C++ function), 594, 1103, 1774, 1876  
sh::ui::web::WebStoreProfileDialog::isCloseableByUser (C++ function), 595, 1103, 1775, 1877  
sh::ui::web::WebStoreProfileDialog::isInitiated (C++ function), 595, 1103, 1775, 1877  
sh::ui::web::WebStoreProfileDialog::manager (C++ function), 595, 1104, 1775, 1877  
sh::ui::web::WebStoreProfileDialog::profileName (C++ function), 594, 1103, 1774, 1876  
sh::ui::web::WebStoreProfileDialog::setCloseableByUser (C++ function), 594, 1103, 1775, 1877  
sh::ui::web::WebStoreProfileDialog::setDialogProperty (C++ function), 594, 1103, 1774, 1876  
sh::ui::web::WebStoreProfileDialog::toJson (C++ function), 595, 1103, 1775, 1877  
sh::ui::web::WebStoreProfileDialog::triggerDialogEvent (C++ function), 594, 1103, 1774, 1876  
sh::ui::web::WebStoreProfileDialog::toJson (C++ function), 595, 1103, 1775, 1877  
sh::ui::web::WebStoreProfileDialog::triggerDialogEvent (C++ function), 594, 1103, 1775, 1877  
sh::ui::web::WebStoreProfileDialog::waitClosed (C++ function), 595, 1104, 1775, 1877  
sh::ui::web::WebStoreProfileDialog::wasAccepted (C++ function), 595, 1104, 1775, 1877  
sh::ui::web::WebStoreProfileDialog::wasClosed (C++ function), 595, 1104, 1775, 1877  
sh::ui::web::WebStoreProfileDialog::wasClosed (C++ function), 595, 1104, 1775, 1877  
sh::ui::web::WebStoreProfileDialog::WebStoreProfileDialog (C++ function), 594, 1103, 1774, 1876  
sh::ui::web::WebStoreProfileDialog::withSubDirectories (C++ function), 594, 1103, 1774, 1876  
sh::ui::web::WebTuningDialog (C++ class), 596, 1104, 1776, 1878  
sh::ui::web::WebTuningDialog::\_cfgvalues (C++ member), 597, 1105, 1777, 1879  
sh::ui::web::WebTuningDialog::\_cfgvaluescondition (C++ member), 597, 1105, 1777, 1879  
sh::ui::web::WebTuningDialog::\_cfgvaluesinitiated (C++ member), 597, 1105, 1777, 1879  
sh::ui::web::WebTuningDialog::\_cfgvaluesmutex (C++ member), 597, 1105, 1777, 1879  
sh::ui::web::WebTuningDialog::categoryToName (C++ function), 597, 1106, 1777, 1879  
sh::ui::web::WebTuningDialog::close (C++ function), 596, 1105, 1777, 1879  
sh::ui::web::WebTuningDialog::cmd\_\_change\_configurationvalue (C++ function), 597, 1105, 1777, 1879  
sh::ui::web::WebTuningDialog::cmd\_\_get\_configurationvalues (C++ function), 597, 1105, 1777, 1879  
sh::ui::web::WebTuningDialog::configurationValueToJsonObject (C++ function), 597, 1106, 1777, 1879  
sh::ui::web::WebTuningDialog::dialogId (C++ function), 596, 1105, 1776, 1878  
sh::ui::web::WebTuningDialog::dialogProperty (C++ function), 596, 1104, 1776, 1878  
sh::ui::web::WebTuningDialog::dialogResult (C++ function), 596, 1104, 1776, 1878  
sh::ui::web::WebTuningDialog::isCloseableByUser (C++ function), 596, 1105, 1776, 1878  
sh::ui::web::WebTuningDialog::isInitiated (C++ function), 596, 1105, 1776, 1878  
sh::ui::web::WebTuningDialog::manager (C++ function), 597, 1105, 1777, 1879  
sh::ui::web::WebTuningDialog::setCloseableByUser (C++ function), 596, 1104, 1776, 1878  
sh::ui::web::WebTuningDialog::setDialogProperty (C++ function), 596, 1104, 1776, 1878  
sh::ui::web::WebTuningDialog::toJson (C++ function), 596, 1105, 1776, 1878  
sh::ui::web::WebTuningDialog::triggerDialogEvent (C++ function), 596, 1104, 1776, 1878  
sh::ui::web::WebTuningDialog::valueTypeToName (C++ function), 597, 1106, 1777, 1879  
sh::ui::web::WebTuningDialog::waitClosed (C++ function), 596, 1105, 1776, 1878  
sh::ui::web::WebTuningDialog::wasAccepted (C++ function), 596, 1105, 1776, 1878  
sh::ui::web::WebTuningDialog::wasClosed (C++ function), 597, 1105, 1777, 1879  
sh::ui::web::WebTuningDialog::WebTuningDialog (C++ function), 596, 1104, 1776, 1878