parsley - Manual

Release 3.3.2800

Josef Hahn

Jan 24, 2021

CONTENTS

1 License **1**

2 About **3**

3 Up-to-date? **5**

4 Maturity **7**

5 Dependencies **9**

6 Introduction **11**
- 6.1 First Steps . . . 11
- 6.2 Synchronization Model . . . 12

7 Using Parsley In Windowless Mode **15**

8 Graphical Configuration **17**

9 Configuration Model **21**
- 9.1 Synchronization Tasks . . . 21
- 9.2 Loggers . . . 22
- 9.3 Includes . . . 22
- 9.4 Custom Aspects . . . 23
- 9.5 Python Imports . . . 23

10 Filesystem Conflicts **25**

11 Reporting **27**

12 Customizing Parsley **29**
- 12.1 General Workflow Overview . . . 29
- 12.2 Customizable Parts . . . 32
- 12.3 High Level Customization . . . 32
- 12.4 Python Imports . . . 32

13 Appendix: Command Line **33**

14 Appendix: Installation **35**
- 14.1 Source Code Archive . . . 35

15 API reference **37**
- 15.1 parsley package . . . 37

15.2 parsley_gui module 112
15.3 parsley_infssync_manageconflicts module 112
15.4 parsley_infssync_manageconflicts_gui module 112
15.5 parsley_report_gui module 112

Python Module Index **113**

Index **115**

CHAPTER

ONE

LICENSE

parsley is written by Josef Hahn under the terms of the AGPLv3.

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

CHAPTER

TWO

ABOUT

Parsley keeps a configured set of places in file systems in sync.

Features:

- Keeps configured file system places in sync (local and ssh)
- Robust infrastructure with working retry and error handling
- Customizable behavior with the availability to add additional program logic for various situations
- Optional 'move to sink mode': always moves all files from the source to a sink and so keep the source empty
- Has a mechanism for metadata synchronization (tags, rating, . . .)
- Can be used stand-alone or embedded in other tools with a flexible and extensible api
- Rich graphical interface for configuration and for executing synchronization
- Graphical interface for manually resolving conflicts that occurred in a synchronization run
- Designed for being driven by a scheduled task (a.k.a. cronjob), which executes a background command (e.g. each minute)
- In background mode: Own handling of synchronization intervals (independent of the interval for the scheduled task)

CHAPTER

THREE

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/parsley and check that. You are currently reading the manual for version 3.3.2800.

CHAPTER

FOUR

MATURITY

parsley is in production-stable state.

CHAPTER

FIVE

DEPENDENCIES

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

 *Python 3.7*, required

 *Typical GNU/Linux Desktop*, recommended

sshfs, optional

 *python3-pyxattr*, optional

 *Gtk 3 and PyGObject*, recommended : for user interfaces.

 *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.

 *all files in /_meta*, included : if not mentioned otherwise, Copyright 2015 Josef Hahn under license CC BY-SA 3.0 license.

CHAPTER

SIX

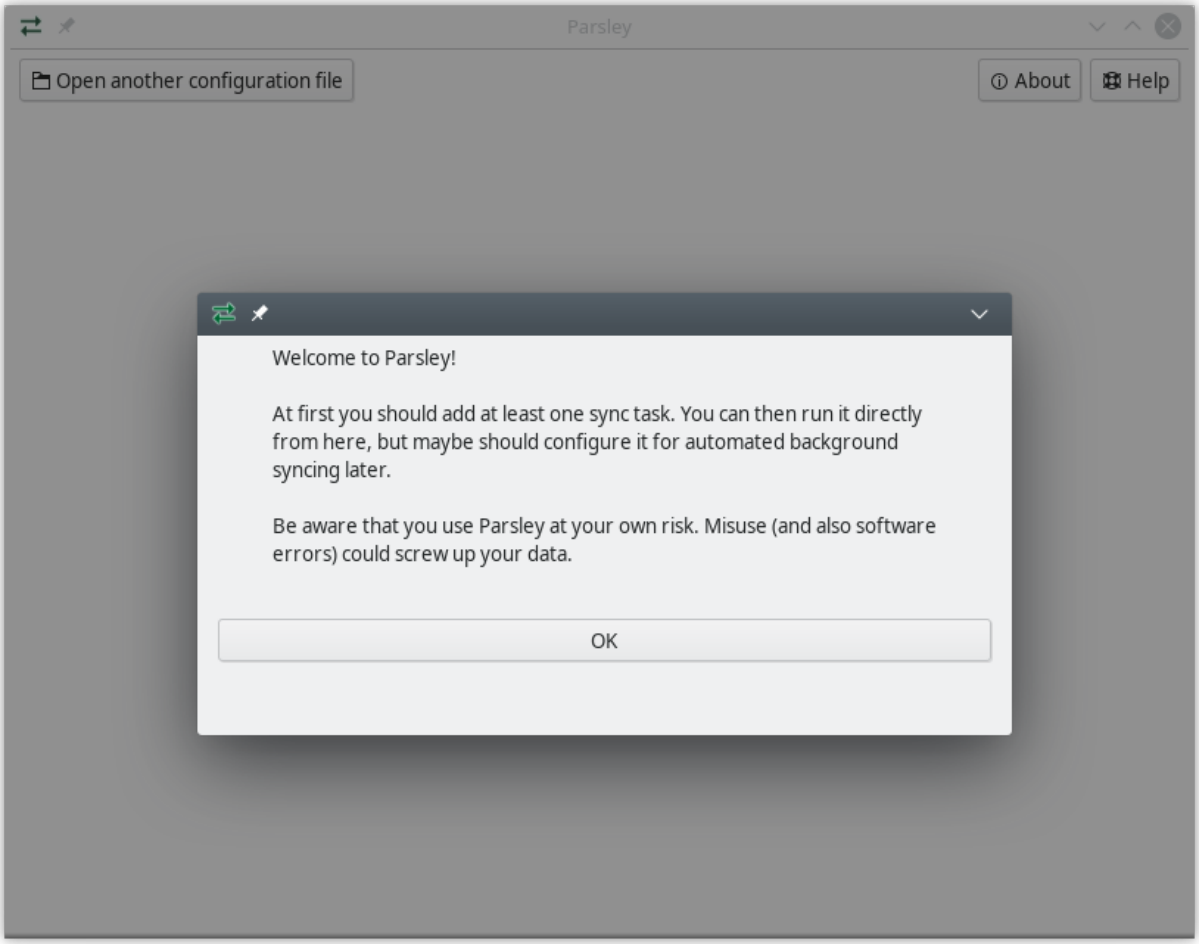
INTRODUCTION

Please read how to make Parsley ready for the first steps in *Appendix: Installation*.

6.1 First Steps

Parsley can run in graphical mode as well as in background mode. The latter one is designed for automated file synchronization, e.g. by periodical runs. It is described in *a later section*.

For the first steps, start Parsley in graphical mode. Either find the appropriate entry in your start menu or call `parsley_gui`.



It automatically creates an empty configuration file in `~/.parsley/parsley.xml`. All changes you do in the Parsley window will eventually be stored there.

It is easy to add a new synchronization now and let it run. This is great for configuration and for testing. For productive usage, it is recommended to run parsley in windowless mode. See the next chapter for more details.

You can set up your configuration entirely from the Parsley window after start. Read more about the user interface in *Graphical Configuration* and also about the *Configuration Model*.

6.2 Synchronization Model

This section describes how Parsley proceeds in order to keep two filesystem locations synchronized, i.e. storing the same files. It can only explain how a usual configuration behaves. This behavior can be marginally or completely different once *Custom Aspects* or other advanced features are used.

In order to keep two filesystem places in sync, Parsley traverses those filesystem trees and operates per file:

- If a file is equally existing on both sides, it proceeds to the next.
- If a file only exists in one place, it decides whether to delete the one or to clone the other (by evaluating if the file existed before).
- If a file exists on both sides with different content, it checks which one is the fresher one (by evaluating the files' modification times) and updates the other. If both files changed since the last run, a *filesystem conflict* occurred.

It synchronizes the file content and a few metadata.

CHAPTER

SEVEN

USING PARSLEY IN WINDOWLESS MODE

The Parsley core component does not have any graphical interface but is designed to run completely in background without any user intervention.

The Parsley command line tool `parsley` runs the synchronization processes this way. It reads your Parsley configuration file (or another other) and executes the synchronizations defined there.

The windowless mode is the recommended day-to-day mode. Parsley is to be called in background in regular intervals (which should be short since parsley has an own interval logic). Add such a line to your `crontab` (or a similar 'scheduled task' in Windows or use whatever your OS provides for executing a command every few minutes):

```
*/3 * * * * /usr/lib/parsley/parsley.py --sync ALL
```

Please adapt the Parsley path to your system.

If you want to use another location for the configuration, create it at first by calling `parsley --createconfig --configfile /some/other/dir/parsley.xml` and adding ` –configfile /some/other/dir/parsley.xml` to the crontab command. You can of course open and edit the resulting `/some/other/dir/parsley.xml` in the graphical configuration tool as well. Read *Appendix: Command Line* for more command line parameters.

You should also add a value `interval` on each for your synchronizations and set it to a time interval like `20m`. Otherwise they all actually will run every time Parsley is called.

For details about return values of the Parsley command line tool, see *parsley.runtime.returnvalue.ReturnValue*.

CHAPTER

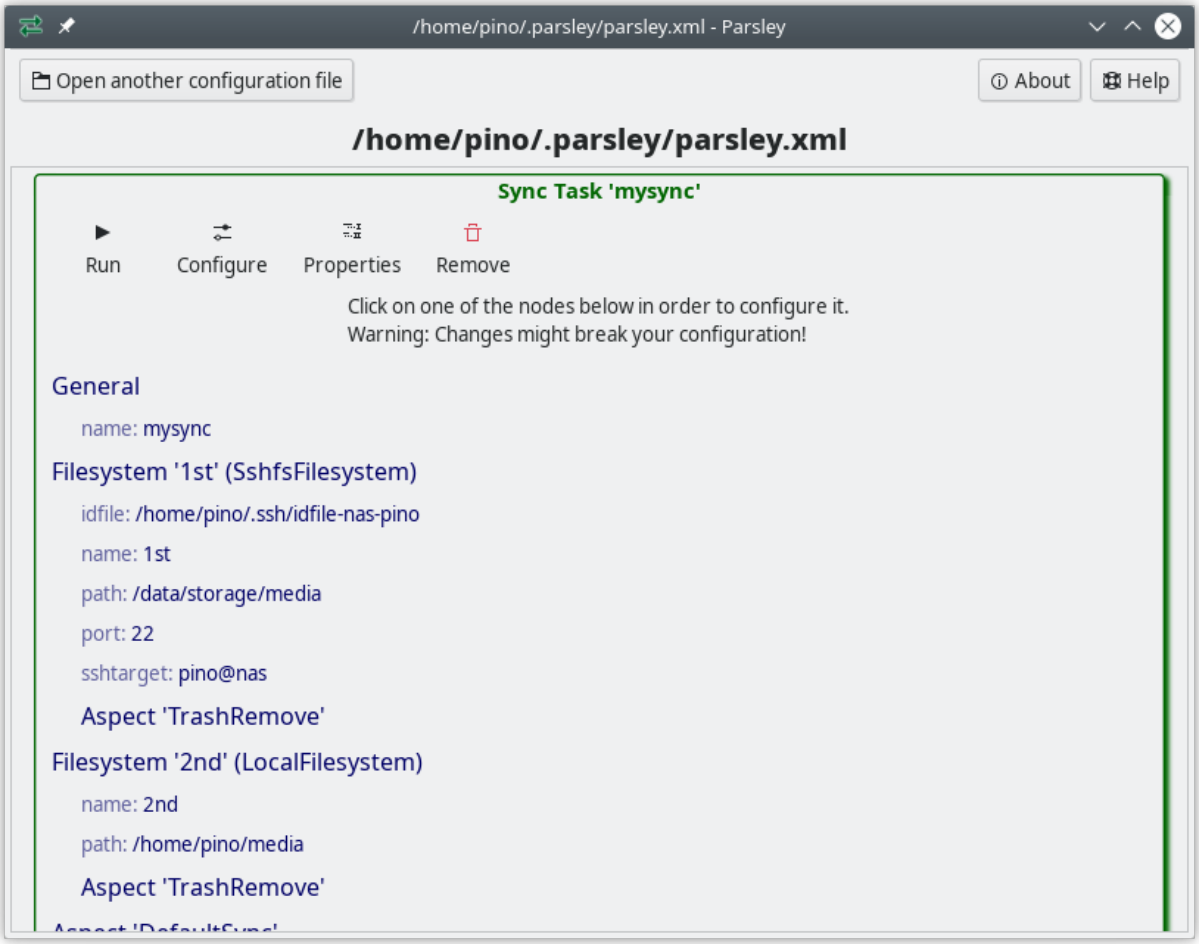
EIGHT

GRAPHICAL CONFIGURATION

The Parsley main window gives an overview about all synchronizations and other stuff you have configured in the currently opened configuration file. You can open different ones at any time. Call `parsley --createconfig --configfile /some/other/place/foo.xml` on command line for creating a fresh Parsley configuration file in some place.

A fresh configuration is mostly empty and has just a Logger configured, so you get output information when syncing. You should not remove it, unless you really want to get rid of that its output. The user interface offers the 'Add item' action, which adds new parts to your configuration. A 'Sync Task' is what you typically would add in the beginning, while the other stuff is for more advanced cases and beyond this manual.

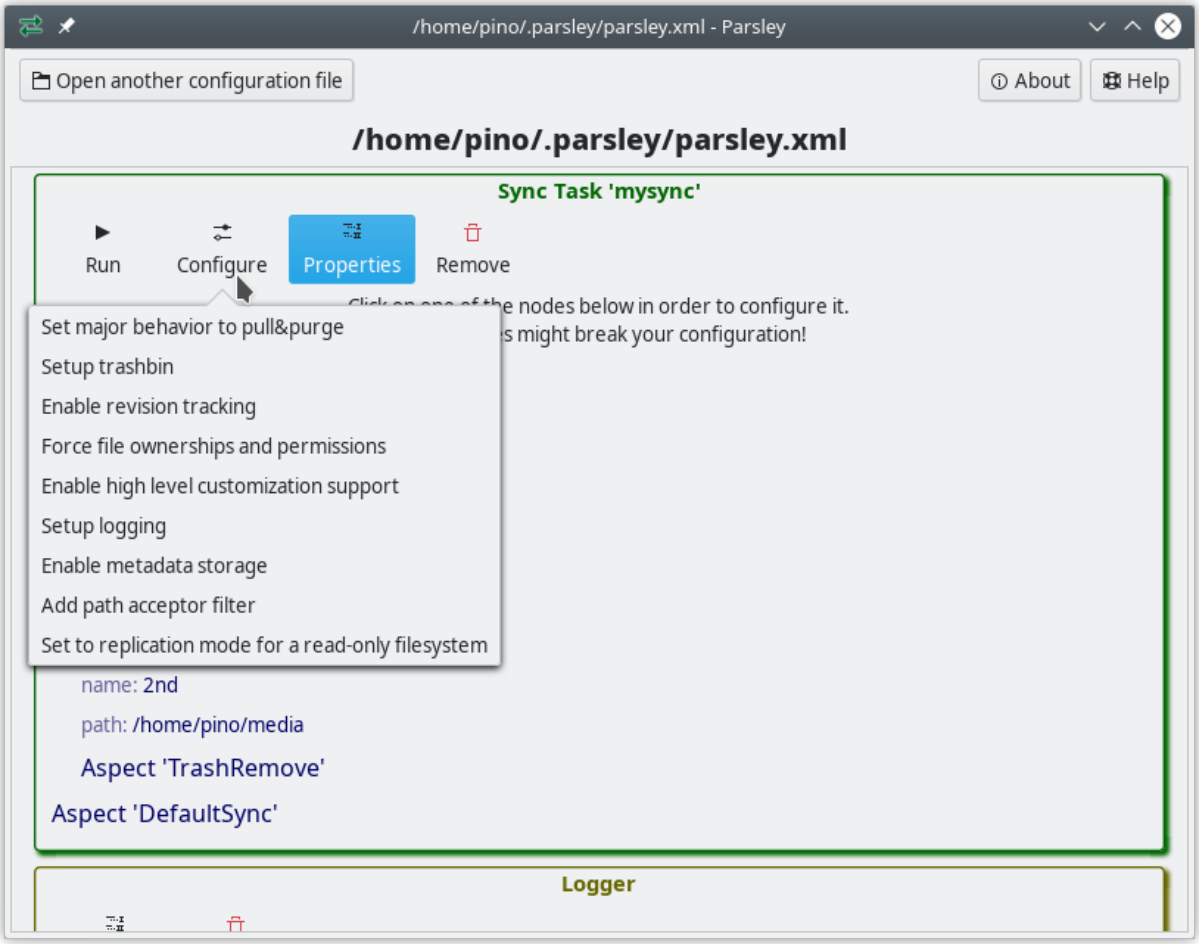
Once you have created a new sync task (sometimes also called: 'sync configuration', just 'sync', 'synchronization', ...) in Parsley, you will see it in the main window. You can 'Configure' it, and clicking on its 'Properties' button offers all configuration details in a direct way (that should be used with care).



Each action modifies your configuration in some way. The following explains the configuration of a sync task in detail.

Configure

This opens a menu of possible configuration changes.



Those guided changes do not need deeper knowledge about Parsley and often no reading of documentation.

Remove sync

Removes an entire synchronization task.

General, Preparation, Filesystem or Aspect level: Add parameter

Adds a name/value pair to the section.

Only add new parameters that are actually allowed for the underlying data structure. The *Configuration Model* section will explain more details.

General or Filesystem level: Add aspect

Adds a new aspect to the filesystem or to the general section.

Only add new aspects that actually exist (either in `parsley.aspect` or custom ones). The *Configuration Model* section will explain more details.

Filesystem level: Add preparation

Adds a new preparation to the general section.

This is only used for exotic cases.

Parameter, Preparation or Aspect level: Remove

Removes a section entirely.

Both potentially changes the behavior of this synchronization, so you should know what you are doing! Do not remove values that are required for the underlying data structure. The *Configuration Model* section will explain more details.

Parameter level: Change value

Changes a value.

You should know about the allowed input values before you change something.

Preparation, Aspect or Filesystem level: Change type

Changes the type of a preparation, aspect or filesystem.

This casts a filesystem or aspect to another type. Only choose a new type that actually exists (`parsley.filesystem`, `parsley.aspect` or custom ones). After you changed the type, it might be required to add and remove some values according to what the new data structure expects to get. The *Configuration Model* section will explain more details.

CHAPTER

NINE

CONFIGURATION MODEL

Beyond some configuration wizards, large parts of the graphical configuration directly reflect structures from the configuration file. So, for advanced usage, knowledge about the configuration file format is often required.

The configuration of Parsley is (at least if you do not use it embedded in your own Python program) written in xml files. As default, the file `parsley.xml` in your home directory will be used. You may use a different one with the `--configfile` command line parameter.

A Parsley configuration file contains configuration objects listed as sub nodes in the root node `parsleyconfig`. There are different kinds of objects (e.g. loggers, synchronization tasks, ...), which can be seen in the different tag names in the xml.

Hint: For most values in parsley configuration files, notations may contain references like *$FOO*, which get replaced by that particular operating system environment variable.

9.1 Synchronization Tasks

Synchronization task configurations are the most interesting ones in most situations. They specify a pair of filesystem locations and a lot of optional additional stuff. The Parsley engine will run the specified synchronization tasks as they are configured here.

A synchronization task configuration - using a `sync` tag in xml - is by far the most complex kind. An example can be seen in `_meta/parsley.xml.example`. The following shows and explains the formal structure:

```
<?xml version="1.0" ?>
<parsleyconfig>
    <sync name="example" interval="5m" ...>
        <fs type="..." name="foo" ...>
            <aspect type="..." .../>
            <aspect type="..." .../>
        </fs>
        <fs type="..." name="bar" ...>
            <aspect type="..." .../>
            <aspect type="..." .../>
        </fs>
        <aspect type="..." .../>
        <aspect type="..." .../>
        <preparation type="..." .../>
        <preparation type="..." .../>
    </sync>
    ...
```

</parsleyconfig>

A `sync` tag contains a name (e.g. used in log messages) and a synchronization interval. For more options, see *parsley.syncengine.sync.Sync*.

It contains two `fs` tags, which specify filesystem locations. They also have a name each and specify a filesystem location (local, ssh, or whatever is supported) for synchronization. See *parsley.filesystem* for existing implementations.

Each `fs` tag may contain `aspect` tags. They control the synchronization behavior, since an aspect is a bunch of small program pieces that react on different events in the synchronization workflow. The complete synchronization functionality, even the builtin one, is part of aspects. See *parsley.aspect* for existing implementations.

The `sync` tag may also contain `aspect` tags directly. Those aspects apply to all filesystems. It is the same as copying those tags into each `fs` tag.

It may also contain `preparation` tags. They specify some actions that must take place before the synchronization can take place. Mounting external filesystems is a very common example for this kind of actions. See *parsley.preparation* for existing implementations.

9.2 Loggers

Loggers can output parsley log messages in some way to some target. The configuration of one `logger` follows this structure:

```
<?xml version="1.0" ?>
<parsleyconfig>
    <logger minseverity="debug" maxseverity="debug" ...>
        <out type="..." ... />
        <formatter type="..." .../>
    </logger>
    ...
</parsleyconfig>
```

It specifies a minimum and maximum severity that shall be logged (see *parsley.logger.logger.Severity*). It also contains a `formatter` configuration for an instance of *parsley.logger.formatter.abstractlogformat.Logformat* (formats the log message) and a `out` configuration for a *parsley.logger.loggerout.abstractloggerout.Loggerout* (actually does the output).

An example can be seen in `_meta/parsley.xml.example`.

See *parsley.logger* for all available functionality.

9.3 Includes

A configuration file can include other ones. Those files have the same structure as primary configuration files and must be complete, including the `parsleyconfig` xml root node.

A configuration file can be included with `include`, this way:

```
<?xml version="1.0" ?>
<parsleyconfig>
    <include path="./some_other_file.xml"/>
    ...
</parsleyconfig>
```

9.4 Custom Aspects

A configuration file can bring the implementation for a custom aspect, which can then be used in some sync task configurations. Those implementations are provided in a `customaspect`:

```
<?xml version="1.0" ?>
<parsleyconfig>
    <sync ...>
        ...
        <aspect type="DoSomething" />
    </sync>
    <customaspect name="DoSomething">
from parsley.aspect import *
class DoSomething(Aspect):
    def __init__(self):
        Aspect.__init__(self)
    @hook("", "", "", event=SyncEvent.UpdateDir_Prepare)
    def sleepwhilebeginupdatedir(self, ea, fs, ctrl):
        do_something()
    </customaspect>
</parsleyconfig>
```

A custom aspect can be used for executing some custom code in some situations, e.g. for keeping EXIF tags of JPEG files clean or doing something with metadata tags of other media files.

Read the *Customizing Parsley* section for details about implementing a custom aspect.

9.5 Python Imports

Python Imports are used for customization. It allows importing arbitrary Python class or functions from any available module, so you can refer to it at other places. The configuration of one `pythonimport` follows this structure:

```
<?xml version="1.0" ?>
<parsleyconfig>
    <pythonimport importfrom="my.mo.du.le.MyFilesystem" to="MyFilesystem" />
    ...
    <sync ...>
        <fs type="MyFilesystem" ...>
        ...
    </sync>
</parsleyconfig>
```

While `importfrom` must be a full name pointing to a Python object that is importable, `to` is just a bare name without dots!

CHAPTER

TEN

FILESYSTEM CONFLICTS

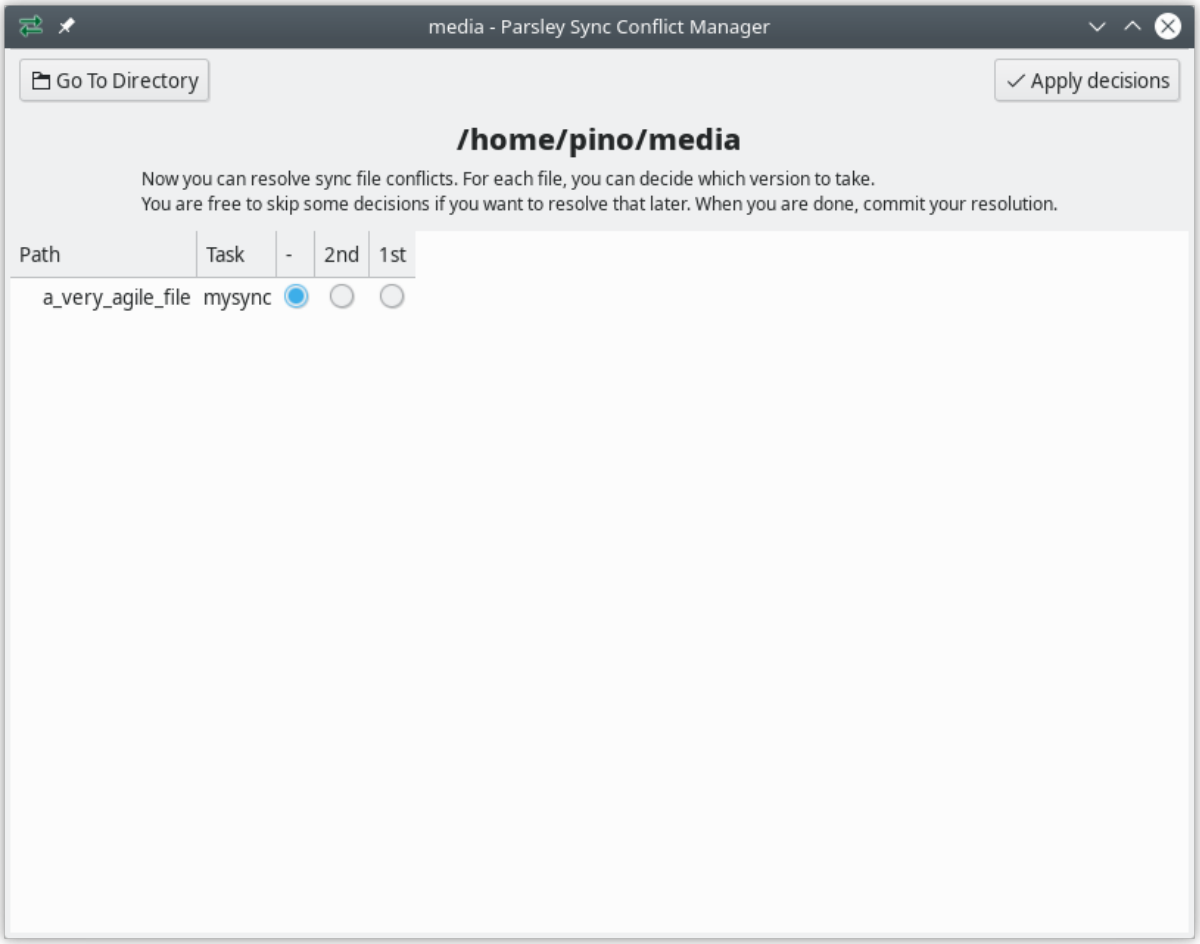
Parsley might encounter conflicts in synchronization runs. Those are situations with incompatible changes of one items on both sides. Those situations must be cleared manually by the user.

Since the Parsley synchronization engine runs decoupled from user intervention, it just stores information about this conflict, so the user can decide later how to resolve it.

There is a graphical user interface available for manually resolving filesystem conflicts. Find it in your start menu, in the Parsley overview, or execute `parsley_infssync_manageconflicts_gui`.

There is also a command line tool available as `parsley_infssync_manageconflicts`. It is designed for scripted usage. Just start it for getting further details.

After a conflict occurs, those tools can be used to manually resolve each issue. These tools store a conflict resolution information, which is applied when the synchronization task runs the next time. They do not execute any synchronization action directly.


media - Parsley Sync Conflict Manager
Go To Directory
Apply decisions
/home/pino/media
Now you can resolve sync file conflicts. For each file, you can decide which version to take.
You are free to skip some decisions if you want to resolve that later. When you are done, commit your resolution.
Path
Task
-
2nd
1st
a_very_agile_file
mysync

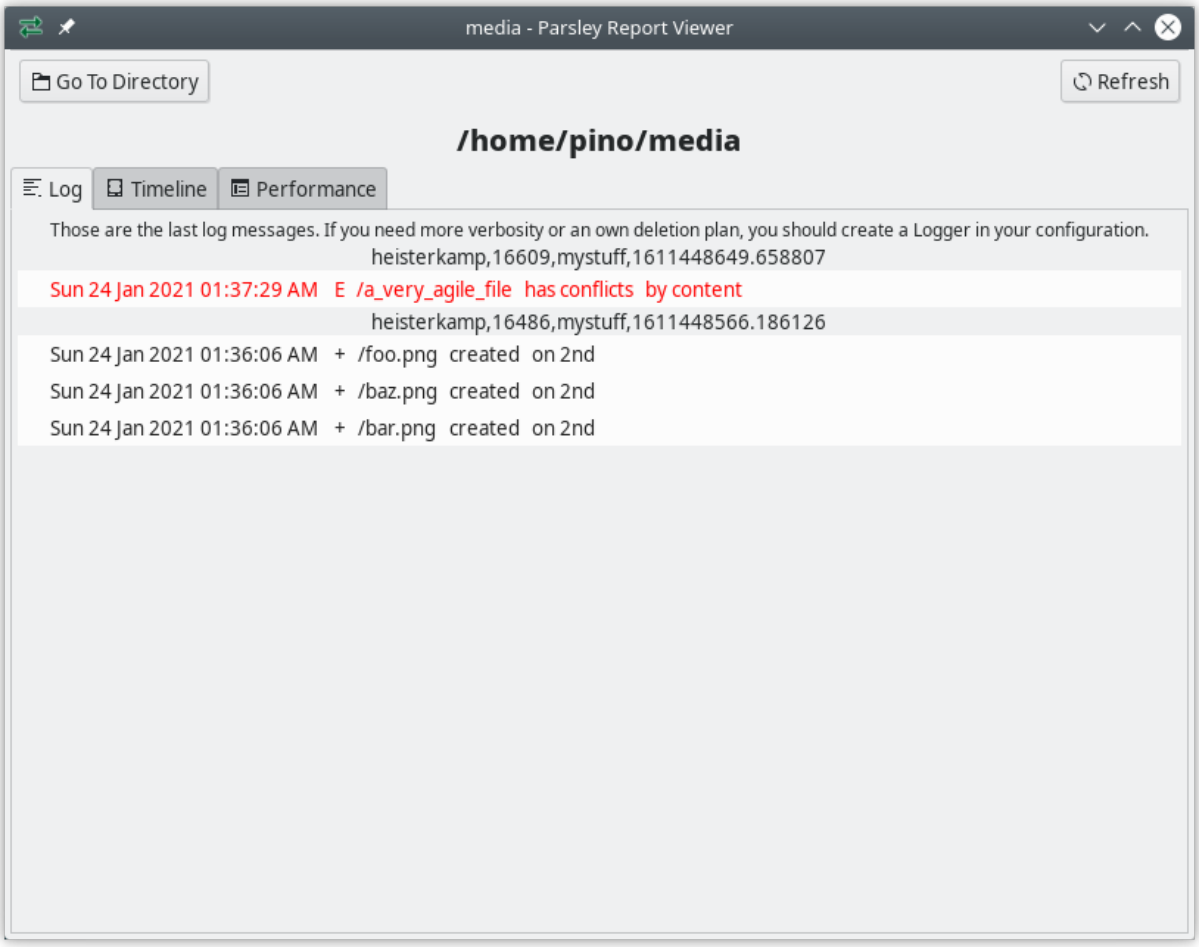
CHAPTER

ELEVEN

REPORTING

Parsley collects some process and telemetry information while it executes your synchronization tasks. This includes the execution logs for each run and performance data.

There is a convenient user interface for inspecting those data. Find it in your start menu, in the Parsley overview, or execute `parsley_report_gui`.



CHAPTER

TWELVE

CUSTOMIZING PARSLEY

12.1 General Workflow Overview

The following describes how the inner parts of parsley work together. This knowledge is very helpful for planning and implementing a customization.

For each `<sync>` in your configuration, the parsley engine will create and configure one instance of *parsley.syncengine.sync.Sync*. If it is not skipped (e.g. because it was already executed less time ago than the `interval` defines), the engine tries to prepare the execution.

Preparing a synchronization means activating all `<preparation>` specified for this synchronization task. This can mount filesystems or whatever is needed for bringing an environment in place, which is required for the actual synchronization to run. Each `preparation` is one instance of a subclass of *parsley.preparation.abstractpreparation.Preparation*, which provides the implementation for activating a preparation before synchronization, for deactivating it afterwards and for status checks.

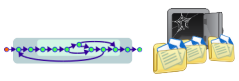
If the synchronization task is successfully prepared, the actual sync operation begins. The sync operation iterates over whatever it can find in your filesystems and just triggers certain events. Without anything more, it would not do anything (and, technically, it would not even iterate that much - but let's forget that for now). The complete synchronization behavior comes with a bunch of small pieces of program code that react on those events. Even the builtin parsley synchronization behavior is implemented as aspects (which also means that you can completely get rid of it by not listing those aspects in your sync configuration).

Those event handlers are added to the pipe by means of some `<aspect>`. Each specified aspect (either within one `<filesystem>` or directly within the `<sync>`) brings an instance of a subclass of *parsley.aspect.abstractaspect.Aspect*, which registers one or more event handlers to the synchronization pipe. One aspect typically implements a certain piece of behavior (which often needs to react on more than only one event).

Intermediate summary: A synchronization task itself is not an interesting thing. It will just fly over your files doing nothing. It can be enriched with some preceding or subsequent actions by means of a `preparation`. But all the interesting synchronization behavior comes with event handlers. Those event handlers are bundled in some `aspect`.

The following description gives a more detailed overview of how parsley would fly over your filesystem and which events are triggered on that flight (i.e. which junction points exist, where aspects can hook in for own logic). For most events, additional information is available in the developer documentation.

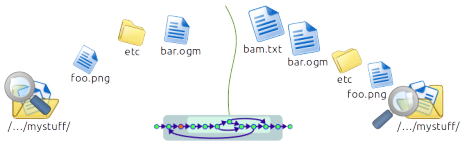
- At the very beginning of the synchronization run (directly after it is prepared), *parsley.syncengine.common.SyncEvent.BeginSync* is triggered.



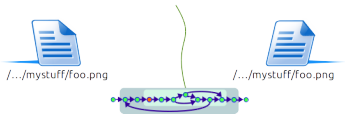
- Afterwards it starts the synchronization of the root directory. Synchronization of a directory (the root one and all the other ones) executes those steps:

- *parsley.syncengine.common.SyncEvent.UpdateDir_Prepare* is triggered for preparing the directory synchronization.

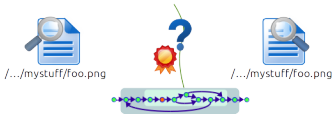
- *parsley.syncengine.common.SyncEvent.UpdateDir_ListDir* is triggered for collecting a list of all direct child entries (files, subdirectories, ...) in that directory.



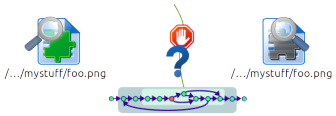
- Afterwards, parsley iterates over all the listed child entries that were listed and synchronizes this entry. Each entry synchronization executes those steps:
  - *parsley.syncengine.common.SyncEvent.UpdateItem_BeforeElectMaster* is triggered for preparing the next step.



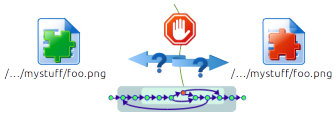
  - *parsley.syncengine.common.SyncEvent.UpdateItem_ElectMaster* is triggered for electing the master filesystem. This is the filesystem that contains the 'right' version of that item. All the other filesystems are meant to get updated according to this version. The result of the election could be to skip all the other steps for this child entirely. It can also select a non-existing location (which typically leads to deletion later on).



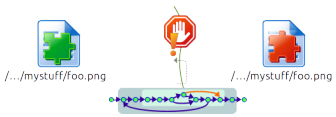
  - *parsley.syncengine.common.SyncEvent.UpdateItem_CheckConflicts* is triggered for checking if a conflict exists between the master filesystem and any other one.



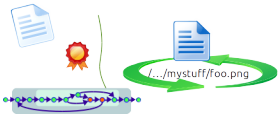
  - *parsley.syncengine.common.SyncEvent.UpdateItem_ResolveConflicts* is triggered for resolving those conflicts, if conflicts appeared.



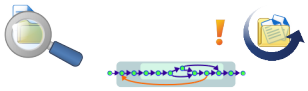
  - *parsley.syncengine.common.SyncEvent.UpdateItem_SkippedDueConflicts* is triggered if conflicts appeared and could not be resolved. In that situation, after triggering this event, some of the next steps are skipped and processing resumes at `UpdateItem_AfterUpdate` (see below).



* *parsley.syncengine.common.SyncEvent.UpdateItem_Update_Prepare* is triggered for preparing the actual updating.

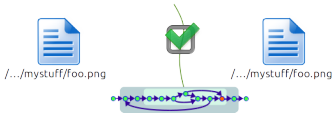


* *parsley.syncengine.common.SyncEvent.UpdateItem_Update_ExistsInMaster* is triggered for actually updating an entry, if it exists in the master filesystem (typically in order to copy it to the other filesystems). If the entry is a directory, this also starts synchronizing this directory by means of the workflow described here (beginning above). This is realized by *parsley.aspect.baseinfrastructure.BaseInfrastructure*, which is always implicitely included in each synchronization configuration.

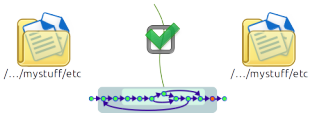


* *parsley.syncengine.common.SyncEvent.UpdateItem_Update_NotExistsInMaster* is triggered for actually updating an entry, if it does not exist in the master filesystem (typically in order to remove it from the other filesystems).

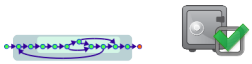
* *parsley.syncengine.common.SyncEvent.UpdateItem_AfterUpdate* is triggered after the entry was synchronized.



– *parsley.syncengine.common.SyncEvent.UpdateDir_AfterUpdate* is triggered after the directory was synchronized.



- At the end of the synchronization, *parsley.syncengine.common.SyncEvent.EndSync* is triggered.



Whenever an event occurs, all registered event handlers are executed. Each event handler execution takes place on top (or: is associated with) one of your specified filesystems. Each event handler typically does its work in that particular filesystem. If an aspect that provides a certain event handler is specified in a `filesystem`, it will be executed exactly for that filesystem. If it is directly specified in the `sync`, it will be executed once for each filesystem.

There is a mechanism for ordering the execution of the event handlers within one event. Please read *parsley.syncengine.sync.Sync.executeevent()* for more details about the ordering and more about the internals.

12.2 Customizable Parts

If you want to override or enhance some parts of the default behavior, read the following parts:

- *parsley.aspect.abstractaspect.Aspect* is the base class of your implementation if you want to develop a part of logical behavior, like 'copy a file to somewhere in some situations'. Inspect the sources in *parsley.aspect* for lots of practical examples. Read also about how to put your *Custom Aspects* to a configuration.
- *parsley.filesystem.abstractfilesystem.Filesystem* is the base class of your own filesystem implementation, which allows usage of a filesystem that is neither the local one nor another support one. Insect the sources in *parsley.filesystem* for examples.
- *parsley.preparation.abstractpreparation.Preparation* is the base class for sync preparations, which will be enabled before the synchronization runs and disabled afterwards by the Parsley engine. One typical use case is mounting a remote filesystem. Inspect the sources in *parsley.preparation* for examples.

12.3 High Level Customization

The *parsley.aspect.highlevelcustomization.HighLevelCustomization* aspect allows to include Python code pieces from somewhere in the synchronization directory tree at defined places into the synchronization behavior.

This is very convenient for including some automation tasks, e.g. automatically converting some kinds of files whenever the appear or reacting in any other custom way to filesystem updates.

Include this aspect to your configuration in order to use this feature. The graphical interface has a guide for it as well.

12.4 Python Imports

It is possible to implement own stuff in external Python modules and use those classes and functions from within the configuration (e.g. as a different filesystem type). Specify a *Python Import* for such a class or function.

CHAPTER

THIRTEEN

APPENDIX: COMMAND LINE

The parsley command-line tool understands this syntax:

```
parsley [options]*
```

Options can be some of the following:

- `--sync [syncname]` : Runs the synchronization of `syncname`. Use `ALL` for `syncname` in order to run all synchronizations.
- `--listsyncs` : Lists all available synchronization configurations.
- `--datadir [dirpath]` : Uses `dirpath` as control data storage directory instead of the default one (`~/.parsley`).
- `--configfile [configfile]` : Uses configuration file `configfile` instead of the default one (`~/.parsley/parsley.xml`).
- `--createconfig` : Creates a fresh configuration file (can be combined with `datadir`).
- `--forcesync [syncname]` : Marks the synchronization `syncname` for forceful synchronization, even if the time interval is not elapsed yet. Can be used more than once.
- `--lock [pid]` : Just acquires the lock, so no other synchronization run will actually do anything until you `unlock`. Used for backup. Without a pid, the lock must be unlocked and refreshed every 10 minutes!
- `--unlock` : Releases a lock acquired with `--lock`.

CHAPTER

FOURTEEN

APPENDIX: INSTALLATION

Install Parsley 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.

14.1 Source Code Archive

Use the source code archive as fallback. Extract it to a location that 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.

It is highly recommended to also establish a command line link or alias for `parsley/parsley.py` so you just have to type `parsley` (`ln -s ...parsley/parsley.py /usr/local/bin/parsley` on Unix or any other operating system specific way). Do the same for `parsley/parsley_gui.py`, `parsley/parsley_infssync_manageconflicts.py` and `parsley/parsley_infssync_manageconflicts_gui.py`. This is according to what the installation packages do and required for executing the exact same commands as used in this manual (otherwise you must substitute the full name for the short command names in this manual).

CHAPTER

FIFTEEN

API REFERENCE

15.1 parsley package

15.1.1 Subpackages

parsley.aspect package

Submodules

parsley.aspect.abstractaspect module

Aspects control the behavior of *parsley.syncengine.sync.Sync* configurations.

class parsley.aspect.abstractaspect.**Aspect**
Bases: object

Abstract base class for aspect implementations that build the actual behavior of a parsley.syncengine.sync.Sync synchronization implementation.

For implementing custom synchronization behavior, implement a subclass of this class and write some aspect hook methods. An aspect hook method is just a method that follows a certain signature and is registered by means of *parsley.aspect.hook* (hook()). A minimalist aspect looks like this:

@verbatim class DoSomething(parsley.aspect.abstractaspect.Aspect):

def __init__(self): parsley.aspect.abstractaspect.Aspect.__init__(self)

@parsley.aspect.hook("", "", "", event=parsley.syncengine.common.SyncEvent.UpdateDir_Prepare)
def sleepwhilebeginupdatedir(self, ctx, filesystem): # this signature must be followed!

do_something()

@endverbatim

A sync configuration can be enhanced by this functionality by adding this aspect to one or more filesystems or globally (which enhances all filesystems).

A hook method is called on top of a certain filesystem whenever a certain event occurs or a certain state in the sync controller is reached. These are actually two different things, even if it sounds similar. It essentially means different modes of the parsley.aspect.hook function (read there for more).

Each call to an aspect hook method gets this parameters:

- *ctx*: a parsley.syncengine.syncruntime.SyncEventRuntime, which holds many flow control information and intermediate data for the execution of one event. It also has some control flow methods.

- *filesystem*: the parsley.filesystem.abstractfilesystem.Filesystem the method is executed on top (it should execute whatever needs to be done for particul filesystem).

Find some function decorators in this module for usage with aspect hook methods, e.g. for some control flow.

parsley.aspect.abstractaspect.**execute_only_for_master_fs**()
Used on aspect hook methods for executing the function body only for the master filesystem.

Creates a function decorator for usage with methods of parsley.aspect.abstractaspect.Aspect subclasses.

Return type Callable

parsley.aspect.abstractaspect.**execute_only_for_master_fs_filetype**(**types*)
Used on aspect hook methods for executing the function body only if the file has certain types in the master filesystem.

Creates a function decorator for usage with methods of parsley.aspect.abstractaspect.Aspect subclasses.

Parameters **types** (*str*) – All arguments are parsley.syncengine.common.EntryType.

Return type Callable

parsley.aspect.abstractaspect.**execute_only_for_non_master_fs**()
Used on aspect hook methods for executing the function body only for non-master filesystems.

Creates a function decorator for usage with methods of parsley.aspect.abstractaspect.Aspect subclasses.

Return type Callable

parsley.aspect.abstractaspect.**execute_only_for_slave_fs_filetype**(**types*)
Used on aspect hook methods for executing the function body only if the file has certain types in the argument filesystem.

Creates a function decorator for usage with methods of parsley.aspect.abstractaspect.Aspect subclasses.

Parameters **types** (*str*) – All arguments are parsley.syncengine.common.EntryType.

Return type Callable

parsley.aspect.abstractaspect.**execute_only_if_not_already_maximally_elected**()
Used on aspect hook methods for executing the function body only if there is not already an election with maximum key.

Creates a function decorator for usage with methods of parsley.aspect.abstractaspect.Aspect subclasses.

Return type Callable

parsley.aspect.abstractaspect.**execute_only_if_not_update_set_skipped**()
Used on aspect hook methods for executing the function body only if the update is not marked as skipped.

Creates a function decorator for usage with methods of parsley.aspect.abstractaspect.Aspect subclasses.

Return type Callable

parsley.aspect.abstractaspect.**hook**(*after*, *provides*, *before*, *event*)
Used for registering a function as parsley.syncengine.sync.Sync hook for implementating parts of Parsley synchronization behavior (those are sometimes called 'aspect hook methods').

Creates a function decorator for usage with methods of parsley.aspect.abstractaspect.Aspect subclasses.

The parameters control the alignment (i.e. the ordering of all aspect hook methods) and to which event to assign the method.

after, *provides* and *before* control the alignment. Please read parsley.syncengine.sync.Sync for details about ordering.

event specifies the event for which this method is to be hooked.

See parsley.syncengine.common.SyncEvent for all events. Read parsley.syncengine.sync.Sync about how all the hook methods are actually executed. Read the user manual about how to directly add custom code into your Parsley configuration files.

Parameters

- **after** (*str*) –
- **provides** (*str*) –
- **before** (*str*) –
- **event** (*str*) –

Return type Callable

parsley.aspect.applypathacceptor module

class parsley.aspect.applypathacceptor.**ApplyPathAcceptor**(*function*)
Bases: *parsley.aspect.abstractaspect.Aspect*

Filters out files/dirs that should be excluded from syncing. It works by providing a path acceptor function.

Parameters **function** – The acceptor function. it gets the path as parameter and must return *True* for accepting it. Warning: This parameter is a string that will be evaluated as Python expression (so usage from config files is easy)! The path argument is available as *path*. Example: *path[2]=='a'*. The filesystem is available as *fs*.

applypathacceptor_applypathacceptor(*ctx*, *filesystem*)
Skips this entry if it is rejected by the path acceptor.

parsley.aspect.baseinfrastructure module

class parsley.aspect.baseinfrastructure.**BaseInfrastructure**
Bases: *parsley.aspect.abstractaspect.Aspect*

Provides very basic infrastructure functionality like the workflow described in parsley.syncengine.sync.Sync. It is implicitly available and must never be added explicitely to a configuration.

baseinfrastructure_update_directory(*ctx*, *filesystem*)
Triggers the directory synchronization if the item is a directory.

parsley.aspect.collectinformation module

class parsley.aspect.collectinformation.**CollectInformation**
Bases: *parsley.aspect.abstractaspect.Aspect*

Used for populating the *._filelists_curr* lists.

collectinformation_collectinfo(*ctx*, *filesystem*)
Collects information (type, size, . . .) for this entry from the filesystem and stores it to _filelists_curr.

parsley.aspect.conflicts module

class parsley.aspect.conflicts.**DetectTypeConflicts**
Bases: *parsley.aspect.abstractaspect.Aspect*

Detects type conflicts between two filesystems (file vs directory, for example). Part of parsley.aspect.defaults.DefaultSync.

detect_type_conflict(*ctx*, *filesystem*)
Detects a type conflict to master.

class parsley.aspect.conflicts.**ResolveConflictsByHint**
Bases: *parsley.aspect.abstractaspect.Aspect*

Resolve conflicts together with TrackConflicts. Part of parsley.aspect.defaults.DefaultSync.

resolveconflicts_byhint(*ctx*, *filesystem*)
Resolves a type conflict by a conflict resolution hint stored in the control files.

class parsley.aspect.conflicts.**TrackConflicts**
Bases: *parsley.aspect.abstractaspect.Aspect*

Tracks conflict for handling outside of parsley (interactively in a gui for example).

For a conflict in file p/f in a sync task t, a control file *conflicts/t/p/f* is created. The conflict can be resolved outside of parsley in an arbitrary way by modifying this file in a certain way (you find examples somewhere in the parsley sources).

Part of parsley.aspect.defaults.DefaultSync.

cleanup_conflicts_dir(*ctx*, *filesystem*)
Cleans up old entries.

prepare_trackconflicts(*ctx*, *filesystem*)
Prepares stuff.

trackconflicts_trackskipped_skipconflict(*ctx*, *filesystem*)
Stores information about the conflict for resolving.

parsley.aspect.defaultiteration module

class parsley.aspect.defaultiteration.**DefaultIteration**
Bases: *parsley.aspect.abstractaspect.Aspect*

Lists child elements via the filesystem. Part of parsley.aspect.defaults.DefaultBase.

defaultiteration_listdir(*ctx*, *filesystem*)
Adds all the child entries to the list (as returned by the underlying filesystem).

parsley.aspect.defaults module

class parsley.aspect.defaults.**DefaultBase**

Bases: *parsley.aspect.defaults.DefaultBaseBare*, *parsley.aspect.electmaster.ElectMasterFileByMtime*, *parsley.aspect.electmaster.ElectMasterLinkByTargetHistory*

Mid-basic part of default behavior for a plain synchronization. Part of parsley.aspect.defaults.DefaultSync.

Parameters **function** – The acceptor function. it gets the path as parameter and must return *True* for accepting it.

Warning: This parameter is a string that will be evaluated as Python expression (so usage from config files is easy)! The path argument is available as *path*. Example: *path[2]=='a'*. The filesystem is available as *fs*.

class parsley.aspect.defaults.**DefaultBaseBare**

Bases: *parsley.aspect.defaultiteration.DefaultIteration*, *parsley.aspect.collectinformation.CollectInformation*, *parsley.aspect.applypathacceptor.ApplyPathAcceptor*, *parsley.aspect.readmefile.ReadmeFile*, *parsley.aspect.volumepathbreadcrumb.VolumePathBreadcrumb*, *parsley.aspect.volumesyncreport.VolumeSyncReport*

Very basic part of default behavior for a plain synchronization. Part of parsley.aspect.defaults.DefaultBase.

Parameters **function** – The acceptor function. it gets the path as parameter and must return *True* for accepting it.

Warning: This parameter is a string that will be evaluated as Python expression (so usage from config files is easy)! The path argument is available as *path*. Example: *path[2]=='a'*. The filesystem is available as *fs*.

class parsley.aspect.defaults.**DefaultSync**

Bases: *parsley.aspect.defaults.DefaultBase*, *parsley.aspect.update.DefaultUpdateItems*, *parsley.aspect.conflicts.DetectTypeConflicts*, *parsley.aspect.conflicts.TrackConflicts*, *parsley.aspect.conflicts.ResolveConflictsByHint*, *parsley.aspect.remove.DetectRemoval*, *parsley.aspect.remove.DefaultRemoveDirs*

Default behavior for a plain synchronization.

Parameters **function** – The acceptor function. it gets the path as parameter and must return *True* for accepting it.

Warning: This parameter is a string that will be evaluated as Python expression (so usage from config files is easy)! The path argument is available as *path*. Example: *path[2]=='a'*. The filesystem is available as *fs*.

class parsley.aspect.defaults.**PullAndPurgeSyncSink**

Bases: *parsley.aspect.defaults.DefaultBaseBare*, *parsley.aspect.update.DefaultUpdateItems*, *parsley.aspect.remove.CleanupTrashBin*

Default behavior for a sink filesystem in a pull-and-purge configuration.

Parameters **function** – The acceptor function. it gets the path as parameter and must return *True* for accepting it.

Warning: This parameter is a string that will be evaluated as Python expression (so usage from config files is easy)! The path argument is available as *path*. Example: *path[2]=='a'*. The filesystem is available as *fs*.

ppsyncsink_electifnowhereelse(*ctx*, *filesystem*)

Elects the sink filesystem if the item exists here ... - by a minimal key, if it is a file, so it gets elected if there are no others - by a maximum key, if it is a directory, so the sync can go deeper

ppsyncsink_renameexistingtonew(*ctx*, *filesystem*)
Renames an already existing entry to some new name.

class parsley.aspect.defaults.**PullAndPurgeSyncSource**
Bases: *parsley.aspect.defaults.DefaultBase*

Default behavior for a source filesystem in a pull-and-purge configuration.

Parameters **function** – The acceptor function. it gets the path as parameter and must return *True* for accepting it.

Warning: This parameter is a string that will be evaluated as Python expression (so usage from config files is easy)! The path argument is available as *path*. Example: *path[2]=='a'*. The filesystem is available as *fs*.

ppsyncsource_removefromsource(*ctx*, *filesystem*)
Removes the file in the source filesystem.

parsley.aspect.electmaster module

class parsley.aspect.electmaster.**ElectMasterFileByMtime**
Bases: *parsley.aspect.abstractaspect.Aspect*

Elects a master file by mtime. Default master election strategy for files. Part of parsley.aspect.defaults.DefaultBase.

electfilebymtime(*ctx*, *filesystem*)
Returns the mtime of this entry as election key.

class parsley.aspect.electmaster.**ElectMasterLinkByTargetHistory**
Bases: *parsley.aspect.abstractaspect.Aspect*

Elects a master link by changes in history. Default master election strategy for links. Part of parsley.aspect.defaults.DefaultBase.

electlinkbyparam(*ctx*, *filesystem*)
Returns 0 if the link remained unchanged since last sync, or 1 otherwise as election key.

parsley.aspect.highlevelcustomization module

class parsley.aspect.highlevelcustomization.**HighLevelCustomization**
Bases: *parsley.aspect.abstractaspect.Aspect*

Executes high level customization handlers. They are loaded directly from the sync tree, whenever a *.parsley.custom.py* file exists.

class BeforeUpdateEvent(*controller*, *isnew*, *isdeleted*, *ischanged*, *scriptsource*)
Bases: object

CONTINUE_TOUCHED = 'CONTINUE_TOUCHED'

CONTINUE_UNTOUCHED = 'CONTINUE_UNTOUCHED'

continue_touched()

continue_untouched()

exception BeforeUpdateHandlerTerminatedWithoutDecision
Bases: *parsley.exceptions.ParsleyError*

callonbeforeupdate(*ctx*, *filesystem*)

init(*ctx*, *filesystem*)

loadcustomizations(*ctx*, *filesystem*)

unloadcustomizations(*ctx*, *filesystem*)

parsley.aspect.logging module

class parsley.aspect.logging.**Logging**(***, *logcreate='1'*, *logremove='1'*, *logupdate='1'*, *logproblem='1'*)

Bases: *parsley.aspect.abstractaspect.Aspect*

Switchable logging by situation.

_logcreate(*ctx*, *filesystem*)
Logs an entry creation (only hooked if this is activated in aspect configuration).

_logproblem(*ctx*, *filesystem*)
Logs a problem (only hooked if this is activated in aspect configuration).

_logremove(*ctx*, *filesystem*)
Logs an entry removal (only hooked if this is activated in aspect configuration).

_logupdate(*ctx*, *filesystem*)
Logs an entry update (only hooked if this is activated in aspect configuration).

parsley.aspect.metadata module

class parsley.aspect.metadata.**MetadataSynchronization**

Bases: *parsley.aspect.abstractaspect.Aspect*

Synchronizes metadata without a shadow storage (typical for the workstations).

metadata_checkagainstshadow(*ctx*, *filesystem*)
Checks if this entry has a metadata update against the shadow storage. If so, it updates the version numbers (in the in-memory data structure).

metadata_propagatetofilesystem_file(*ctx*, *filesystem*)
Checks if metadata of the entry differ from the filesystem information and, if so, updates the entry metadata in the filesystem (with exactly what it is now stored in-memory).

class parsley.aspect.metadata.**MetadataSynchronizationWithShadow**

Bases: *parsley.aspect.metadata.MetadataSynchronization*

Synchronizes metadata with a shadow storage (typical for the file server).

cleanup_shadow(*ctx*, *filesystem*)
Cleans up orphaned files in shadow metadata storage.

metadata_findhighestshadow(*ctx*, *filesystem*)
Updates *latestmd* as stored in ctx if this filesystem has a higher version in shadow.

metadata_init(*ctx*, *filesystem*)
Some initialization.

metadata_propagatetoshadow_file(*ctx*, *filesystem*)
Checks if metadata of the entry differ from the shadow storage information and, if so, updates the entry metadata in the shadow storage (with exactly what it is now stored in-memory).

exception parsley.aspect.metadata.**SyncMetadataAspectError**

Bases: *parsley.exceptions.ParsleyError*

class parsley.aspect.metadata.**_Metadata**(*version=0*)
Bases: object

static get_metadata_from_file(*filesystem*, *path*, *ignore_removed=False*)

static get_metadata_from_shadow(*filesystem*, *shadowfilesystem*, *path*)

getdata(*k*, *d=None*)

getkeys()

static metadata_differs(*md1*, *md2*)

putdata(*k*, *v*)

static remove_shadow(*shadowfilesystem*, *path*, *isdir*)

static set_metadata_to_file(*filesystem*, *path*, *md*, *ignore_removed=False*)

static set_metadata_to_shadow(*filesystem*, *shadowfilesystem*, *path*, *md*)

static set_metadataversion_to_file(*filesystem*, *path*, *md*, *ignore_removed=False*)

class parsley.aspect.metadata.**_MetadataStruct**
Bases: object

parsley.aspect.monitorfilechanges module

class parsley.aspect.monitorfilechanges.**MonitorFileChanges**
Bases: *parsley.aspect.abstractaspect.Aspect*

Tries to establish a proxy on the other end of an ssh filesystem connection that monitors file changes. If changes appear, a sync is triggered for the next possible moment. It only works on parsley.filesystem.sshfs.SshfsFilesystem's.

_beforeelectmaster(*ctx*, *filesystem*)
Remembers which files have been seen, so we can catch just notifications for them (the other ones are not interesting).

_beginsync(*ctx*, *filesystem*)
Establishes the proxy.

_closesync(*ctx*, *filesystem*)
Stops the filesseenwriter thread (and more).

filesseenwriterthread(*sync*, *filesystem*, *notifyonly_storage*, *filesseen*, *filesseen_lock*)

static monitorproxy(*sshtarget*, *port*, *idfile*, *proxypath*, *rvolpath*, *lastsuccfile*, *notifyonlylist*, *asynclogpath*, *options*)

parsley.aspect.permissions module

class parsley.aspect.permissions.**ApplyPermissions**(*user*, *group*, *fileaddperms=''*, *filesubtractperms=''*, *diraddperms=''*, *dirsubtractperms=''*)
Bases: *parsley.aspect.abstractaspect.Aspect*

Applies the specified file system permissions to all entries processed by parsley. Warning: Do not use this feature in security relevant contexts. It's not guaranteed that all aspects follow this.

Parameters

- **user** – Either "#uid" or user name of the new owner user.
- **group** – Either "#gid" or group name of the new owner group.
- **fileaddperms** – Numerical mask of permissions that will be added to each synchronized file.
- **filesubtractperms** – Numerical mask of permissions that will be subtracted from each synchronized file.
- **diraddperms** – Numerical mask of permissions that will be added to each synchronized directory.
- **dirsubtractperms** – Numerical mask of permissions that will be subtracted from each synchronized directory.

static _setperms(*path*, *uid*, *gid*, *addperms*, *subtractperms*)

applypermissions_setdirperms(*ctx*, *filesystem*)
Sets the file permissions of a dir entry to what is configured in the aspect.

applypermissions_setfileperms(*ctx*, *filesystem*)
Sets the file permissions of a file entry to what is configured in the aspect.

parsley.aspect.readmefile module

class parsley.aspect.readmefile.**ReadmeFile**
Bases: *parsley.aspect.abstractaspect.Aspect*

Writes a readme file into the on-volume parsley control directory.

readmefile_writereadme(*ctx*, *filesystem*)
Writes the readme.

parsley.aspect.remove module

class parsley.aspect.remove.**CleanupTrashBin**(*trashdelay='7d'*)
Bases: *parsley.aspect.abstractaspect.Aspect*

Cleans up the trash bin. Part of parsley.aspect.remove.DefaultRemove and parsley.aspect.remove.TrashRemove.

cleanup_init(*ctx*, *filesystem*)
Some initialization.

cleanup_trashbin(*ctx*, *filesystem*)
Cleans up files in the trashbin control directory that fulfill certain conditions.

class parsley.aspect.remove.**DefaultRemove**
Bases: *parsley.aspect.remove.RemoveOrphanedDirs*, *parsley.aspect.remove.CleanupTrashBin*

Default removal strategy without a trashbin.

defaultremove_file_link(*ctx*, *filesystem*)
Removes the file or link from the filesystem.

class parsley.aspect.remove.**DefaultRemoveDirs**
Bases: *parsley.aspect.abstractaspect.Aspect*

Removes directories via the filesystem. Part of parsley.aspect.defaults.DefaultSync.

defaultremovedirs_removedir(*ctx*, *filesystem*)
Removes a directory (recursively or safely) in the filesystem.

class parsley.aspect.remove.**DetectRemoval**
Bases: *parsley.aspect.abstractaspect.Aspect*

Detects removed entries. Part of parsley.aspect.defaults.DefaultSync.

static _mtimes_equal(*t1*, *t2*, *precise*)

detectremoval_elect(*ctx*, *filesystem*)
Detects if an entry is to be removed and if so, returns the current time as election key.

class parsley.aspect.remove.**RemoveOrphanedDirs**
Bases: *parsley.aspect.abstractaspect.Aspect*

Removes empty orphaned directories. Part of parsley.aspect.remove.DefaultRemove and parsley.aspect.remove.TrashRemove.

check_orphaned_dirs(*ctx*, *filesystem*)

remove_orphaned_dirs(*ctx*, *filesystem*)
Removes a directory in the filesystem if it is empty and was removed in another filesystem.

class parsley.aspect.remove.**TrashRemove**(*trashdelay='7d'*)
Bases: *parsley.aspect.remove.RemoveOrphanedDirs*, *parsley.aspect.remove.CleanupTrashBin*

Removal strategy with a trashbin.

trashremove_file_link(*ctx*, *filesystem*)
Moves a file or link to the trash bin.

parsley.aspect.revisiontracking module

class parsley.aspect.revisiontracking.**RevisionTracking**(*number_unarchived_revisions=3*, *number_revisions_per_archive=20*)
Bases: *parsley.aspect.abstractaspect.Aspect*

Keeps all versions of all files that parsley has seen into a revision storage in the control directory.

Parameters

- **number_unarchived_revisions** – Number of revisions per file to be stored directly, not inside an archive file.
- **number_revisions_per_archive** – Number of revisions per file to be stored in one archive, before the next archive file begins.

static _updatefile_helper(*ctx*, *filesystem*, *path*, *number_unarchived_revisions*, *number_revisions_per_archive*)

revisiontracking_init(*ctx*, *filesystem*)
Some initialization.

revisiontracking_store1(*ctx*, *filesystem*)
Checks if the version with the current mtime of this entry is already stored in the version history, and if not, do so.

revisiontracking_store2(*ctx*, *filesystem*)
Checks if the version with the current mtime of this entry is already stored in the version history, and if not, do so.

parsley.aspect.update module

class parsley.aspect.update.**DefaultUpdateItems**
Bases: *parsley.aspect.abstractaspect.Aspect*

Default handling for updating non-directory items (i.e. files, links). Part of parsley.aspect.defaults.DefaultSync.

defaultupdateitem_detectandskipupdateconflict(*ctx*, *filesystem*)
Checks if an entry stays in an update conflict (by mtime/param comparisons) and if so, mark and skip the entry.

defaultupdateitem_skipidentical(*ctx*, *filesystem*)
Skips an entry if they have identical mtime (or link target, for links) in both filesystems.

defaultupdateitem_skipidentical_aux(*ctx*, *filesystem*)

defaultupdateitem_update(*ctx*, *filesystem*)
Updates the entry in the non-master filesystems.

resolveconflicts_cleanupbeforenewdir(*ctx*, *filesystem*)
Forcefully remove files/links for paths that begin a directory sync (so they must be directories).

parsley.aspect.volumepathbreadcrumb module

class parsley.aspect.volumepathbreadcrumb.**VolumePathBreadcrumb**
Bases: *parsley.aspect.abstractaspect.Aspect*

Writes a file to the data directory that helps finding the sync volumes after runtime.

volumepathbreadcrumb_write(*ctx*, *filesystem*)
Writes the breadcrumb.

parsley.aspect.volumesyncreport module

class parsley.aspect.volumesyncreport.**VolumeSyncReport**
Bases: *parsley.aspect.abstractaspect.Aspect*

Writes some report data to the volume.

syncvolumereport_write(*ctx*, *filesystem*)
Writes the report.

Module contents

Aspects control the behavior of *parsley.syncengine.sync.Sync* configurations. Each feature should be encapsulated as one subclass of *parsley.aspect.abstractaspect.Aspect* that hooks some event handlers into the processing chain (so-called aspect hook methods).

This module also contains some function decorators for usage with aspect hook methods.

parsley.config package

Submodules

parsley.config.config module

Parsley configuration handling.

class parsley.config.config.**ParsleyConfiguration**(*environment=None*)
Bases: object

Representation of a parsley configuration. This class can parse a configuration from xml and can generate it. It also has some modification methods and internal methods for generating actual objects from the data.

Parameters **environment** (*Optional[Dict[str, str]]*) –

class Aspect(*cfg*, *otype*, ***params*)
Bases: object

Represents configuration for one aspect.

Parameters

- **cfg** (ParsleyConfiguration) – The ParsleyConfiguration instance.
- **otype** (*str*) – The otype name for the new object (i.e. the class name).
- **params** – Additional keyword args are stored as additional config values.

static fromxml(*cfg*, *xentry*)
Generates a new configuration object from an xml piece.
Parameters
- **cfg** (parsley.config.config.ParsleyConfiguration) –
- **xentry** (*xml.etree.ElementTree.Element*) –

Return type *parsley.config.config.ParsleyConfiguration.Aspect*

instantiate(*knowntypes*)
Instantiates a new actual parsley object by this configuration object.
Parameters **knowntypes** (*Dict[str, Type]*) –
Return type *parsley.aspect.abstractaspect.Aspect*

toxml(*xparent*)
Generates an xml piece for this object.
Parameters **xparent** (*xml.etree.ElementTree.Element*) –
Return type None

class CustomAspect(*cfg*, *name*, *code*)
Bases: object

Represents configuration for one custom aspect implementation.

Parameters

- **cfg** (ParsleyConfiguration) – The ParsleyConfiguration instance.
- **name** (*str*) – The new otype name (i.e. class name) for this aspect.
- **code** (*str*) – The implementation source code.

static fromxml(*cfg*, *xentry*)
Generates a new configuration object from an xml piece.
Parameters
- **cfg** (parsley.config.config.ParsleyConfiguration) –

- **xentry** (*xml.etree.ElementTree.Element*) –

Return type *parsley.config.config.ParsleyConfiguration.CustomAspect*

instantiate(*knowntypes*)
Instantiates a new actual parsley object by this configuration object.
Parameters **knowntypes** (*Dict[str, Type]*) –

toxml(*xparent*)
Generates an xml piece for this object.
Parameters **xparent** (*xml.etree.ElementTree.Element*) –
Return type None

class Filesystem(*cfg*, *otype*, *aspects*, ***params*)
Bases: object

Represents configuration for one filesystem.

Parameters

- **cfg** (ParsleyConfiguration) – The ParsleyConfiguration instance.
- **otype** (*str*) – The otype name for the new object (i.e. the class name).
- **aspects** (*List[ParsleyConfiguration.Aspect]*) – A list of ParsleyConfiguration.Aspect.
- **params** – Additional keyword args are stored as additional config values.

static fromxml(*cfg*, *xentry*)
Generates a new configuration object from an xml piece.
Parameters
- **cfg** (parsley.config.config.ParsleyConfiguration) –
- **xentry** (*xml.etree.ElementTree.Element*) –

Return type *parsley.config.config.ParsleyConfiguration.Filesystem*

instantiate(*knowntypes*)
Instantiates a new actual parsley object by this configuration object.
Parameters **knowntypes** (*Dict[str, Type]*) –
Return type *parsley.filesystem.abstractfilesystem.Filesystem*

property name
Return type t.Optional[str]
This property is also settable.

toxml(*xparent*)
Generates an xml piece for this object.
Parameters **xparent** (*xml.etree.ElementTree.Element*) –
Return type None

class Include(*cfg*, *incpath*)
Bases: object

Represents configuration for one config file include.

Parameters

- **cfg** (ParsleyConfiguration) – The ParsleyConfiguration instance.
- **incpath** (*str*) – The path to the parsley configuration file for including.

static fromxml(*cfg*, *xentry*)
Generates a new configuration object from an xml piece.
Parameters

- **cfg** (parsley.config.config.ParsleyConfiguration) –
- **xentry** (*xml.etree.ElementTree.Element*) –

Return type *parsley.config.config.ParsleyConfiguration.Include*

get_absolute_path(*basepath*)

Returns the absolute path to the file to include.

Parameters **basepath** (*str*) –

Return type str

toxml(*xparent*)

Generates an xml piece for this object.

Parameters **xparent** (*xml.etree.ElementTree.Element*) –

Return type None

class LogFormatter(*cfg*, *otype*, ***params*)

Bases: object

Represents configuration for one log formatter.

Parameters

- **cfg** (ParsleyConfiguration) – The ParsleyConfiguration instance.
- **otype** (*str*) – The otype name for the new object (i.e. the class name).
- **params** – Additional keyword args are stored as additional config values.

static fromxml(*cfg*, *xentry*)

Generates a new configuration object from an xml piece.

Parameters

- **cfg** (parsley.config.config.ParsleyConfiguration) –
- **xentry** (*xml.etree.ElementTree.Element*) –

Return type *parsley.config.config.ParsleyConfiguration.LogFormatter*

instantiate(*knowntypes*)

Instantiates a new actual parsley object by this configuration object.

Parameters **knowntypes** (*Dict[str, Type]*) –

Return type *parsley.logger.formatter.abstractlogformat.Logformat*

toxml(*xparent*)

Generates an xml piece for this object.

Parameters **xparent** (*xml.etree.ElementTree.Element*) –

Return type None

class Logger(*cfg*, *minseverity*, *maxseverity*, *loggerout*, *formatter*, *enabled*)

Bases: object

Represents configuration for one logger.

Parameters

- **cfg** (ParsleyConfiguration) – The ParsleyConfiguration instance.
- **minseverity** (*Optional[str]*) – The minimum severity (as parsley.logger.Severity symbol name).
- **maxseverity** (*Optional[str]*) – The maximum severity (as parsley.logger.Severity symbol name).
- **loggerout** (ParsleyConfiguration.Loggerout) – A ParsleyConfiguration.Loggerout.

- **formatter** (ParsleyConfiguration.LogFormatter) – A ParsleyConfiguration.LogFormatter.
- **enabled** (*bool*) – If to enable this logger.

static fromxml(*cfg*, *xentry*)
Generates a new configuration object from an xml piece.
Parameters
- **cfg** (parsley.config.config.ParsleyConfiguration) –
- **xentry** (*xml.etree.ElementTree.Element*) –

Return type *parsley.config.config.ParsleyConfiguration.Logger*

instantiate(*knowntypes*)
Instantiates a new actual parsley object by this configuration object.
Parameters **knowntypes** (*Dict[str, Type]*) –
Return type *parsley.logger.logger.Logger*

toxml(*xparent*)
Generates an xml piece for this object.
Parameters **xparent** (*xml.etree.ElementTree.Element*) –
Return type None

class Loggerout(*cfg*, *otype*, ***params*)
Bases: object

Represents configuration for one logger output.

Parameters
- **cfg** (ParsleyConfiguration) – The ParsleyConfiguration instance.
- **otype** (*str*) – The otype name for the new object (i.e. the class name).
- **params** – Additional keyword args are stored as additional config values.

static fromxml(*cfg*, *xentry*)
Generates a new configuration object from an xml piece.
Parameters
- **cfg** (parsley.config.config.ParsleyConfiguration) –
- **xentry** (*xml.etree.ElementTree.Element*) –

Return type *parsley.config.config.ParsleyConfiguration.Loggerout*

instantiate(*knowntypes*)
Instantiates a new actual parsley object by this configuration object.
Parameters **knowntypes** (*Dict[str, Type]*) –
Return type *parsley.logger.loggerout.abstractloggerout.Loggerout*

toxml(*xparent*)
Generates an xml piece for this object.
Parameters **xparent** (*xml.etree.ElementTree.Element*) –
Return type None

class Preparation(*cfg*, *otype*, ***params*)
Bases: object

Represents configuration for one synchronization task preparation.

Parameters
- **cfg** (ParsleyConfiguration) – The ParsleyConfiguration instance.
- **otype** – The otype name for the new object (i.e. the class name).

- **params** – Additional keyword args are stored as additional config values.

static fromxml(*cfg*, *xentry*)
Generates a new configuration object from an xml piece.
Parameters
- **cfg** (parsley.config.config.ParsleyConfiguration) –
- **xentry** (*xml.etree.ElementTree.Element*) –

Return type *parsley.config.config.ParsleyConfiguration.Preparation*

instantiate(*knowntypes*)
Instantiates a new actual parsley object by this configuration object.
Parameters **knowntypes** (*Dict[str, Type]*) –
Return type *parsley.preparation.abstractpreparation.Preparation*

toxml(*xparent*)
Generates an xml piece for this configuration object.
Parameters **xparent** (*xml.etree.ElementTree.Element*) –
Return type None

class PythonImport(*cfg*, *importfrom*, *to*)
Bases: object

Represents configuration for one python import.

Parameters

- **cfg** (ParsleyConfiguration) – The ParsleyConfiguration instance.
- **importfrom** (*str*) – The full name of the class to import ('package.module.Class').
- **to** (*str*) – The name this class shall have after importing (without dots).

static fromxml(*cfg*, *xentry*)
Generates a new configuration object from an xml piece.
Parameters
- **cfg** (parsley.config.config.ParsleyConfiguration) –
- **xentry** (*xml.etree.ElementTree.Element*) –

Return type *parsley.config.config.ParsleyConfiguration.PythonImport*

instantiate()
Instantiates a new actual parsley object by this configuration object.
Return type Any

toxml(*xparent*)
Generates an xml piece for this object.
Parameters **xparent** (*xml.etree.ElementTree.Element*) –
Return type None

class Sync(*cfg*, *filesystems*, *aspects*, *preparations*, ***params*)
Bases: object

Represents configuration for one synchronization task.

Parameters

- **cfg** (ParsleyConfiguration) – The ParsleyConfiguration instance.
- **filesystems** (*List[ParsleyConfiguration.Filesystem]*) – A list of ParsleyConfiguration.Filesystem.
- **aspects** (*List[ParsleyConfiguration.Aspect]*) – A list of ParsleyConfiguration.Aspect.

- **preparations** (*List[ParsleyConfiguration.Preparation]*) – A list of ParsleyConfiguration.Preparation.
- **params** – Additional keyword args are stored as additional config values.

_aspectlist (*fs=None*)
Returns either the aspect list for *fs*, or the global one if *fs=None*.
Parameters **fs** (*Optional[int]*) –
Return type List[*parsley.config.config.ParsleyConfiguration.Aspect*]

addaspect (*otype*, *fs=None*, ***params*)
Adds a new aspect.
Parameters
- **otype** (*str*) – The otype name for the new aspect (i.e. the class name).
- **fs** (*Optional[int]*) – Adds to this filesystem's aspect list, *None* for global.
- **params** – Additional keyword args are stored as additional config values.

Return type *parsley.config.config.ParsleyConfiguration.Aspect*

static fromxml (*cfg*, *xentry*)
Generates a new configuration object from an xml piece.
Parameters
- **cfg** (parsley.config.config.ParsleyConfiguration) –
- **xentry** (*xml.etree.ElementTree.Element*) –

Return type *parsley.config.config.ParsleyConfiguration.Sync*

getaspects (*otype*, *fs=None*, *everywhere=False*)
Returns all aspects with otype name *otype* (i.e. the class name).
Parameters
- **otype** (*str*) – The otype name.
- **fs** (*Optional[int]*) – Whose aspect list to consider, *None* for the global one.
- **everywhere** (*bool*) – If *True*, all aspect lists are considered; ignores *fs*.

Return type List[*parsley.config.config.ParsleyConfiguration.Aspect*]

hasaspect (*otype*, *fs=None*, *everywhere=False*)
Checks if an aspect with otype name *otype* (i.e. the class name) exists.
Parameters
- **otype** (*str*) – The otype name.
- **fs** (*Optional[int]*) – Whose aspect list to consider, *None* for the global one.
- **everywhere** (*bool*) – If *True*, all aspect lists are considered; ignores *fs*.

Return type bool

instantiate (*knowntypes*)
Instantiates a new actual parsley object by this configuration object.
Parameters **knowntypes** (*Dict[str, Type]*) –
Return type *parsley.syncengine.sync.Sync*

property name
Return type t.Optional[str]
This property is also settable.

removeaspect (*otype*, *fs=None*, *everywhere=False*)
Removes all aspects with otype name *otype* (i.e. the class name).
Parameters
- **otype** (*str*) – The otype name.
- **fs** (*Optional[int]*) – Whose aspect list to consider, *None* for the global one.
- **everywhere** (*bool*) – If *True*, all aspect lists are considered; ignores *fs*.

Return type None

toxml(*xparent*)
Generates an xml piece for this object.
Parameters **xparent** (*xml.etree.ElementTree.Element*) –
Return type None

property customaspects

Return type t.List['ParsleyConfiguration.CustomAspect']

property environment

Return type t.Dict[str, str]

static fromxml(*xmlstring*, *environment=None*)
Generates a ParsleyConfiguration from an xml string.

Parameters

- **xmlstring** (*str*) –
- **environment** (*Optional[Dict[str, str]]*) –

Return type *parsley.config.config.ParsleyConfiguration*

property includes

Return type t.List['ParsleyConfiguration.Include']

property loggers

Return type t.List['ParsleyConfiguration.Logger']

parseparams(*d*)
Interprets a complete dictionary by the same replacement as in ParsleyConfiguration.parsestring.

Parameters **d** (*Dict[str, str]*) –

Return type Dict[str, str]

parsestring(*s*)
" Interprets a string by replacing each *${FOO}* with the value of the os environmental variable *FOO*.

Parameters **s** (*str*) –

Return type str

property pythonimports

Return type t.List['ParsleyConfiguration.PythonImport']

property syncs

Return type t.List['ParsleyConfiguration.Sync']

toxml()
Returns an xml representation for the current configuration.

Return type str

parsley.config.configpiece module

Helpers for Parsley configuration handling.

parsley.config.configpiece.**getbool**(*s*)
If *s* is a *bool*, it is returned directly, otherwise it is parsed from string.

Parameters **s** (*Any*) –

Return type bool

parsley.config.configpiece.**getcallable**(*s*, *argnames*)
If *s* is a *callable*, it is returned directly, otherwise it is parsed from string.

Parameters

- **s** (*Any*) –
- **argnames** (*List[str]*) –

Return type Callable

parsley.config.configpiece.**getlist**(*kwa*, *listname*)
Returns a list from a keyword-args structure either by direct lookup or by flattened lookup (*foo_0*, *foo_1*, *foo_2*, . . . , *foo_n*).

Parameters

- **kwa** (*Dict[str, Optional[Any]]*) –
- **listname** (*str*) –

Return type List[str]

parsley.config.configpiece.**gettimedelta**(*s*)
If *s* is a *datetime.timedelta*, it is returned directly, otherwise it is parsed from string.

Parameters **s** (*Any*) –

Return type datetime.timedelta

Module contents

Parsley configuration.

parsley.exceptions package

Module contents

Basic exception classes.

exception parsley.exceptions.**ConfigurationError**
Bases: *parsley.exceptions.ParsleyError*

exception parsley.exceptions.**EnablingImpossibleExecutionError**
Bases: *parsley.exceptions.ParsleyEngineExecutionError*

exception parsley.exceptions.**ErrorDisablingPreparationExecutionError**
Bases: *parsley.exceptions.ParsleyEngineExecutionError*

exception parsley.exceptions.**ErrorEnablingPreparationExecutionError**
Bases: *parsley.exceptions.ParsleyEngineExecutionError*

exception parsley.exceptions.**ErrorExecutingExecutionError**
Bases: *parsley.exceptions.ParsleyEngineExecutionError*

exception parsley.exceptions.**ErrorGettingPreparationStateExecutionError**
Bases: *parsley.exceptions.ParsleyEngineExecutionError*

exception parsley.exceptions.**ErrorInitializingExecutionError**
Bases: *parsley.exceptions.ParsleyEngineExecutionError*

exception parsley.exceptions.**InvalidCommandLineError**
Bases: *parsley.exceptions.ParsleyError*

exception parsley.exceptions.**ParsleyEngineExecutionError**
Bases: *parsley.exceptions.ParsleyError*

exception parsley.exceptions.**ParsleyError**
Bases: Exception

exception parsley.exceptions.**PreparationDisabledAfterEnablingExecutionError**
Bases: *parsley.exceptions.ParsleyEngineExecutionError*

exception parsley.exceptions.**PreparationEnabledAfterDisablingExecutionError**
Bases: *parsley.exceptions.ParsleyEngineExecutionError*

exception parsley.exceptions.**PreparationEnabledBeforeEnablingExecutionError**
Bases: *parsley.exceptions.ParsleyEngineExecutionError*

exception parsley.exceptions.**ReadInvalidConfigurationError**
Bases: *parsley.exceptions.ConfigurationError*

parsley.filesystem package

Submodules

parsley.filesystem.abstractfilesystem module

Abstract base class for filesystem implementations.

class parsley.filesystem.abstractfilesystem.**Filesystem**(**aspects*, *name*, *checkaliveskeptically='0'*)
Bases: object

Base class for a filesystem implementation.

It's an abstraction layer for all filesystem operations like 'create file' or 'remove x'. See parsley.filesystem for existing implementations.

Parameters

- **aspects** (*parsley.aspect.abstractaspect.Aspect*) –
- **name** (*str*) –
- **checkaliveskeptically** (*str*) –

_create_control_filesystem(*path*)
Creates a control filesystem. *Override this method in custom subclasses.*

Return type Tuple[*parsley.filesystem.abstractfilesystem.Filesystem*, bool]

checkalive()
Ensures that the filesystem is alive and throws an exception otherwise. Call this method whenever you have fetched some infos whose correctness is critical for the further processing.

Return type None

copyfile(*srcpath*, *dstpath*, *verifier=<function Filesystem.<lambda>>*)
Copies a file from the local filesystem to a destination. *Override this method in custom subclasses.*

Parameters

- **srcpath** (*str*) –
- **dstpath** (*str*) –

Return type Tuple[datetime.datetime, int]

createdirs(*path*, *recursive=True*)
Creates a directory (or a chain of directories). *Override this method in custom subclasses.*

Parameters

- **path** (*str*) –
- **recursive** (*bool*) –

Return type List[str]

createlink(*srcpath*, *dstpath*)
Creates a link. *Override this method in custom subclasses.*

Parameters

- **srcpath** (*str*) –
- **dstpath** (*str*) –

Return type None

exists(*path*)
Returns if an item at a given location exists. *Override this method in custom subclasses.*

Parameters **path** (*str*) –

Return type bool

get_control_filesystem(*path=None*)
Returns a parsley.filesystem.abstractfilesystem.Filesystem wrapper to the control directory. This is typically backed within sync volume itself and can be used for storing control information on that (potentially remote) place.

Parameters **path** (*Optional[str]*) –

Return type *parsley.filesystem.abstractfilesystem.Filesystem*

get_parent_filesystem()
If this is a control filesystem, it returns the original one, otherwise *self*.

Return type *parsley.filesystem.abstractfilesystem.Filesystem*

getftype(*path*)
Returns the parsley.syncengine.common.EntryType item type of an item. *Override this method in custom subclasses.*

Parameters **path** (*str*) –

Return type Optional[str]

getfulllocalpath(*path*)
Returns the local filesystem path of a file. Returns *None* if there is representation in the local filesystem. *Override this method in custom subclasses or leave the default implementation.* @note Returning *None* or not implementing it will probably restrict the things parsley can do.

Parameters **path** (*str*) –

Return type Optional[str]

getfulllocalpathorfallback(*path*)
Returns a string representation, containing the local filesystem path of a file or something else.

Parameters **path** (*str*) –

Return type str

getinputstream(*path*)
Returns a binary file stream (like Python's *open* would do) for reading from a path. *Override this method in custom subclasses.*

Parameters **path** (*str*) –

Return type BinaryIO

getlinktarget(*path*)
Returns the link target of a link. *Override this method in custom subclasses.*

Parameters **path** (*str*) –

Return type str

getmtime(*path*)
Returns the time of last modification of a file. *Override this method in custom subclasses.*

Parameters **path** (*str*) –

Return type datetime.datetime

getoutputstream(*path*)
Returns a binary file stream (like Python's *open* would do) for writing to a path. *Override this method in custom subclasses.*

Parameters **path** (*str*) –

Return type BinaryIO

getsize(*path*)
Returns the file size of a file. *Override this method in custom subclasses.*

Parameters **path** (*str*) –

Return type int

getxattrvalue(*path*, *key*)
Returns the extended attribute value for a file and a key. *Override this method in custom subclasses.*

Parameters

- **path** (*str*) –
- **key** (*str*) –

Return type str

initialize(*sync*, *runtime*)
Runs initialization (before preparations are executed). *Override this method in custom subclasses or leave the default implementation.*

Parameters

- **sync** (parsley.syncengine.sync.Sync) –
- **runtime** (parsley.runtime.runtime.RuntimeData) –

Return type None

initialize_late(*sync*, *runtime*)
Runs late initialization (after it is prepared). *Override this method in custom subclasses or leave the default implementation.*

Parameters

- **sync** (parsley.syncengine.sync.Sync) –
- **runtime** (parsley.runtime.runtime.RuntimeData) –

Return type None

is_control_filesystem()
Checks if this is a control filesystem.

Return type bool

is_mtime_precision_fine()
Checks if the filesystem has fine (typically milliseconds) time granularity. *Override this method in custom subclasses or leave the default implementation.*

Return type bool

listdir(*path*)
Returns a list of the names of all items in the given location. *Override this method in custom subclasses.*

Parameters **path** (*str*) –

Return type List[str]

listxattrkeys(*path*)
Returns a list of names of extended attributes stored for a file. *Override this method in custom subclasses.*

Parameters **path** (*str*) –

Return type List[str]

move(*path*, *dst*)
Moves an item. *Override this method in custom subclasses.*

Parameters

- **path** (*str*) –
- **dst** (*str*) –

Return type None

readfromfile(*path*)
Reads the content of a file and returns it. *Override this method in custom subclasses.*

Parameters **path** (*str*) –

Return type bytes

removedir(*path*, *recursive=False*)
Removes a directory. *Override this method in custom subclasses.*

Parameters

- **path** (*str*) –
- **recursive** (*bool*) –

Return type None

removefile(*path*)
Removes a file. *Override this method in custom subclasses.*

Parameters **path** (*str*) –

Return type None

removelink(*path*)
Removes a link. *Override this method in custom subclasses.*

Parameters **path** (*str*) –

Return type None

setxattrvalue(*path*, *key*, *value*)
Sets an extended attribute value on a file. *Override this method in custom subclasses.*

Parameters

- **path** (*str*) –
- **key** (*str*) –
- **value** (*str*) –

Return type None

shutdown(*sync*, *runtime*)
Shuts down the filesystem.

Parameters

- **sync** (parsley.syncengine.sync.Sync) –
- **runtime** (parsley.runtime.runtime.RuntimeData) –

Return type None

translate_path(*path*, *dstfs*)
Translates a path from this filesystem into a path valid for another one. Returns *None* if this translation is not possible.

Parameters

- **path** (*str*) –
- **dstfs** (parsley.filesystem.abstractfilesystem.Filesystem) –

Return type Optional[str]

unsetxattrvalue(*path*, *key*)
Unsets an extended attribute value on a file. *Override this method in custom subclasses.*

Parameters

- **path** (*str*) –
- **key** (*str*) –

Return type None

writetofile(*path*, *content*)
Writes a byte array to a file. *Override this method in custom subclasses.*

Parameters

- **path** (*str*) –
- **content** (*bytes*) –

Return type None

exception parsley.filesystem.abstractfilesystem.**SyncFilesystemError**
Bases: *parsley.exceptions.ParsleyError*

parsley.filesystem.local module

class parsley.filesystem.local.**LocalFilesystem**(**aspects*, *path*, *external_control_directory=None*, ***kwa*)
Bases: *parsley.filesystem.abstractfilesystem.Filesystem*

A location in the local filesystem.

Parameters **external_control_directory** – Absolute path to an external volume storage directory. Use it for read-only filesystems.

_create_control_filesystem(*path*)
Creates a control filesystem. *Override this method in custom subclasses.*

_gettemp()

_translate_remote_time_to_local(*rtime*)

Parameters **rtime** (*datetime.datetime*) –

Return type datetime.datetime

copyfile(*srcpath*, *dstpath*, *verifier=<function LocalFilesystem.<lambda>>*)
Copies a file from the local filesystem to a destination. *Override this method in custom subclasses.*

createdirs(*path*, *recursive=True*)
Creates a directory (or a chain of directories). *Override this method in custom subclasses.*

createlink(*srcpath*, *dstpath*)
Creates a link. *Override this method in custom subclasses.*

exists(*path*)
Returns if an item at a given location exists. *Override this method in custom subclasses.*

getftype(*path*)
Returns the parsley.syncengine.common.EntryType item type of an item. *Override this method in custom subclasses.*

getfulllocalpath(*path*)
Returns the local filesystem path of a file. Returns *None* if there is representation in the local filesystem. *Override this method in custom subclasses or leave the default implementation.* @note Returning *None* or not implementing it will probably restrict the things parsley can do.

getinputstream(*path*)
Returns a binary file stream (like Python's *open* would do) for reading from a path. *Override this method in custom subclasses.*

getlinktarget(*path*)
Returns the link target of a link. *Override this method in custom subclasses.*

getmtime(*path*)
Returns the time of last modification of a file. *Override this method in custom subclasses.*

getoutputstream(*path*)
Returns a binary file stream (like Python's *open* would do) for writing to a path. *Override this method in custom subclasses.*

getsize(*path*)
Returns the file size of a file. *Override this method in custom subclasses.*

getxattrvalue(*path*, *key*)
Returns the extended attribute value for a file and a key. *Override this method in custom subclasses.*

initialize(*sync*, *runtime*)
Runs initialization (before preparations are executed). *Override this method in custom subclasses or leave the default implementation.*

initialize_late(*sync*, *runtime*)
Runs late initialization (after it is prepared). *Override this method in custom subclasses or leave the default implementation.*

is_mtime_precision_fine()
Checks if the filesystem has fine (typically milliseconds) time granularity. *Override this method in custom subclasses or leave the default implementation.*

listdir(*path*)
Returns a list of the names of all items in the given location. *Override this method in custom subclasses.*

listxattrkeys(*path*)
Returns a list of names of extended attributes stored for a file. *Override this method in custom subclasses.*

move(*path*, *dst*)
Moves an item. *Override this method in custom subclasses.*

readfromfile(*path*)
Reads the content of a file and returns it. *Override this method in custom subclasses.*

removedir(*path*, *recursive=False*)
Removes a directory. *Override this method in custom subclasses.*

removefile(*path*)
Removes a file. *Override this method in custom subclasses.*

removelink(*path*)
Removes a link. *Override this method in custom subclasses.*

setxattrvalue(*path*, *key*, *value*)
Sets an extended attribute value on a file. *Override this method in custom subclasses.*

unsetxattrvalue(*path*, *key*)
Unsets an extended attribute value on a file. *Override this method in custom subclasses.*

writetofile(*path*, *content*)
Writes a byte array to a file. *Override this method in custom subclasses.*

exception parsley.filesystem.local.**SyncLocalFilesystemError**
Bases: *parsley.filesystem.abstractfilesystem.SyncFilesystemError*

parsley.filesystem.sshfs module

class parsley.filesystem.sshfs.**SshfsFilesystem**(**aspects*, *name*, *sshtarget*, *path*, *idfile=None*, *password=None*, *port='22'*, *timeout='10s'*, ***kwa*)

Bases: *parsley.filesystem.local.LocalFilesystem*

A location in a (remote) ssh filesystem. Please note that remote ssh filesystems have to be manually prepared before using with parsley. This is needed for some detection features. Please create the directory *.parsley.control* in the root directory of a directory structure, before you try to sync it via ssh with parsley!

Parameters **external_control_directory** – Absolute path to an external volume storage directory. Use it for read-only

filesystems.

class _SshXattrProxy(*sshfs*)

Bases: object

_waitanswer(*endstring*, *additionaltimeout=0*)

request(*command*)

shutdown()

getsshxattrproxy()

getxattrvalue(*path*, *key*)

Returns the extended attribute value for a file and a key. *Override this method in custom subclasses.*

initialize(*sync*, *runtime*)

Runs initialization (before preparations are executed). *Override this method in custom subclasses or leave the default implementation.*

listxattrkeys(*path*)

Returns a list of names of extended attributes stored for a file. *Override this method in custom subclasses.*

setxattrvalue(*path*, *key*, *value*)

Sets an extended attribute value on a file. *Override this method in custom subclasses.*

shutdown(*sync*, *runtime*)

Shuts down the filesystem.

unsetxattrvalue(*path*, *key*)

Unsets an extended attribute value on a file. *Override this method in custom subclasses.*

exception parsley.filesystem.sshfs.**SyncSshFilesystemError**

Bases: *parsley.filesystem.abstractfilesystem.SyncFilesystemError*

exception parsley.filesystem.sshfs.**SyncSshFilesystemRemoteProxyError**

Bases: *parsley.filesystem.sshfs.SyncSshFilesystemError*

Module contents

Implementations of filesystems for use with *parsley.syncengine.sync.Sync*. Each filesystem implementation should be encapsulated as a subclass of *parsley.filesystem.abstractfilesystem.Filesystem*.

parsley.gui package

Subpackages

parsley.gui.app_main package

Submodules

parsley.gui.app_main.configitem module

class parsley.gui.app_main.configitem.**ConfigItem**(*handler*, ***kwargs*)
Bases: gi.overrides.Gtk.Box

Initializer for a GObject based classes with support for property sets through the use of explicit keyword arguments.

Parameters **handler** (ConfigItemHandler) –

_cleanup(*toolbar=True*, *configbody=True*, *body=True*)

_set_showbody_button(*btn*)

_trigger_add_item(*config*, *aligntowidget*)

_trigger_reload()

add_listener(*listener*)

property body_visible

property configobject

init_template(*self*)

lblhead = <gi._gtktemplate.Child object>

pnlbody = <gi._gtktemplate.Child object>

pnlbodyouter = <gi._gtktemplate.Child object>

pnlconfigbody = <gi._gtktemplate.Child object>

refresh()

toolbar = <gi._gtktemplate.Child object>

class parsley.gui.app_main.configitem.**ConfigItemHandler**(*configobject*)
Bases: object

_add_param(*obj*)

_ask_and_remove(*lst*, *obj*)

_change_type(*obj*)

_edit_attr(*obj*, *attrname*, *question*)

_edit_boolattr(*obj*, *attrname*)

_edit_param(*obj*, *paramkey*, *paramvalue*)

_remove_param(*obj*, *paramkey*)

_set_root_config(*cfgfile*, *datadir*)

property _toplevel_window

property configitem

property configobject

abstract get_css_classname()

Return type str

abstract get_header_text()

Return type str

init_ui(*configitem*)

Parameters **configitem** (parsley.gui.app_main.configitem.ConfigItem) –

Return type None

reload()

class parsley.gui.app_main.configitem.**ConfigItemListener**

Bases: object

add_item(*config*, *aligntowidget*)

reload()

class parsley.gui.app_main.configitem.**CustomAspectConfigItemHandler**(*customaspect*, ***kwargs*)

Bases: *parsley.gui.app_main.configitem.ConfigItemHandler*

Parameters **customaspect** (parsley.config.config.ParsleyConfiguration.CustomAspect) –

__do_editcode(*_)

__do_remove(*_)

get_css_classname()

get_header_text()

init_ui(*configitem*)

class parsley.gui.app_main.configitem.**FileIncludeConfigItemHandler**(*fileinclude*, ***kwargs*)

Bases: *parsley.gui.app_main.configitem.ConfigItemHandler*

Parameters **fileinclude** (parsley.config.config.ParsleyConfiguration.Include) –

__do_add(*cfg*, *aligntowidget*)

__do_remove(*_)

get_css_classname()

get_header_text()

init_ui(*configitem*)

class parsley.gui.app_main.configitem.**LoggerConfigItemHandler**(*logger*, ***kwargs*)

Bases: *parsley.gui.app_main.configitem.ConfigItemHandler*

Parameters **logger** (parsley.config.config.ParsleyConfiguration.Logger) –

__do_remove(*_)

__gethandler_edit_attr(*paramkey*, *question*)

__gethandler_edit_boolattr(*paramkey*)

__gethandler_edit_formatter()

__gethandler_edit_formatter_param(*paramkey*, *paramvalue*)

__gethandler_edit_loggerout()

__gethandler_edit_loggerout_param(*paramkey*, *paramvalue*)

get_css_classname()

get_header_text()

init_ui(*configitem*)

class parsley.gui.app_main.configitem.**ObjectConfigNode**(***kwargs*)

Bases: gi.overrides.Gtk.Button

do_get_property(*pspec*)

do_set_property(*pspec*, *value*)

label

class parsley.gui.app_main.configitem.**ParamConfigNode**(***kwargs*)

Bases: gi.overrides.Gtk.Button

do_get_property(*pspec*)

do_set_property(*pspec*, *value*)

key_label

value_label

class parsley.gui.app_main.configitem.**PythonImportConfigItemHandler**(*pythonimport*, ***kwargs*)

Bases: *parsley.gui.app_main.configitem.ConfigItemHandler*

Parameters **pythonimport** (parsley.config.config.ParsleyConfiguration.PythonImport) –

__do_remove(*_)

__gethandler_edit_attr(*paramkey*, *question*)

get_css_classname()

get_header_text()

init_ui(*configitem*)

class parsley.gui.app_main.configitem.**SyncTaskConfigItemHandler**(*synctask*, ***kwargs*)

Bases: *parsley.gui.app_main.configitem.ConfigItemHandler*

Parameters **synctask** (parsley.config.config.ParsleyConfiguration.Sync) –

__call_parsley_tool(*toolname*)

Parameters **toolname** (*str*) –

__do_config(*btn*, *_)

__do_conflicts(*_)

__do_remove(*_)

__do_report(*_)

__do_run(*_)

__gethandler_edit_aspect(*aspect*)

__gethandler_edit_aspect_param(*aspect*, *paramkey*, *paramvalue*)

__gethandler_edit_filesystem(*filesystem*)

__gethandler_edit_filesystem_aspect(*filesystem*, *aspect*)

__gethandler_edit_filesystem_aspect_param(*aspect*, *paramkey*, *paramvalue*)

__gethandler_edit_filesystem_param(*filesystem*, *paramkey*, *paramvalue*)

__gethandler_edit_general()

__gethandler_edit_param(*paramkey*, *paramvalue*)

__gethandler_edit_preparation(*preparation*)

__gethandler_edit_preparation_param(*preparation*, *paramkey*, *paramvalue*)

_add_aspect(*parentlist*)

_add_preparation()

_remove_aspect(*parentlist*, *aspect*)

_remove_preparation(*preparation*)

get_css_classname()

get_header_text()

init_ui(*configitem*)

class parsley.gui.app_main.configitem.**_DummyConfigItemHandler**(*configobject*)

Bases: *parsley.gui.app_main.configitem.ConfigItemHandler*

get_css_classname()

get_header_text()

parsley.gui.app_main.configitem.**configitems_for_config**(*rootconfig_cfgfile*, *rootconfig_datadir*, *config*, *reloadfct*, *additemfct*, *container*)

parsley.gui.app_main.runsyncdialog module

class parsley.gui.app_main.runsyncdialog.**RunSyncDialog**(***kwargs*)
Bases: gi.overrides.Gtk.Dialog

Initializer for a GObject based classes with support for property sets through the use of explicit keyword arguments.

__handle_dialogstate()

append_output(*txt*)

btncancel = <gi._gtktemplate.Child object>

bufferlog = <gi._gtktemplate.Child object>

do_get_property(*pspec*)

do_set_property(*pspec*, *value*)

init_template(*self*)

pnlsyncfailed = <gi._gtktemplate.Child object>

pnlsyncing = <gi._gtktemplate.Child object>

pnlsyncsucceeded = <gi._gtktemplate.Child object>

sync_failed

sync_succeeded

Module contents

The Parsley main user interface.

class parsley.gui.app_main.**App**
Bases: object

__about(*_)

__do_add(*config*, *aligntowidget*)

__do_add_customaspect(*_)

__do_add_fileinclude(*_)

__do_add_logger(*_)

__do_add_pythonimport(*_)

__do_add_synctask(*_)

__help(*_)

__openfile(*filepath*)

__openfiledialog(*_)

_populate()

show()

parsley.gui.app_main.**run**()

parsley.gui.app_manageconflicts package

Module contents

A gui for manually resolving filesystem sync conflicts.

class parsley.gui.app_manageconflicts.**App**
Bases: *parsley.gui.gtk.volumeapp.AbstractVolumeApp*

class Conflict(*path*, *task*, *choices*)
Bases: object

Parameters

- **path** (*str*) –
- **task** (*str*) –
- **choices** (*List[str]*) –

__applydecisions(*_)

classmethod __callmanageconflicts(*workdir*, *params*)

Parameters

- **workdir** (*str*) –
- **params** (*List[str]*) –

Return type List[str]

__getdecisions()

classmethod __listconflicts(*workdir*)

Return type List[*parsley.gui.app_manageconflicts.App.Conflict*]

classmethod __resolveconflict(*workdir*, *path*, *task*, *fsname*)

_has_unsaved_changes()

_populate()

parsley.gui.app_manageconflicts.**run**()

parsley.gui.app_report package

Submodules

parsley.gui.app_report.logpanel module

class parsley.gui.app_report.logpanel.**LogPanel**(**args*, ***kwargs*)
Bases: gi.overrides.Gtk.Box

Initializer for a GObject based classes with support for property sets through the use of explicit keyword arguments.

_populate(*dirpath*)

init_template(*self*)

pnllog = <gi._gtktemplate.Child object>

parsley.gui.app_report.performancepanel module

class parsley.gui.app_report.performancepanel.**PerformancePanel**(***kwargs*)
Bases: gi.overrides.Gtk.Box

Initializer for a GObject based classes with support for property sets through the use of explicit keyword arguments.

__fill_form(*perf_data_available*)

__handle_queryresult(*query_result_available*)

__handle_spinner(_)

__loadresults(*_)

_populate(*dirpath*)

edtaggregation = <gi._gtktemplate.Child object>

edtaxish = <gi._gtktemplate.Child object>

edtaxisv = <gi._gtktemplate.Child object>

edteventhandler = <gi._gtktemplate.Child object>

edtfilesystem = <gi._gtktemplate.Child object>

edtpath = <gi._gtktemplate.Child object>

edtsyncrun = <gi._gtktemplate.Child object>

edtsynctask = <gi._gtktemplate.Child object>

init_template(*self*)

scrollview = <gi._gtktemplate.Child object>

spinner = <gi._gtktemplate.Child object>

tree = <gi._gtktemplate.Child object>

parsley.gui.app_report.timelinepanel module

class parsley.gui.app_report.timelinepanel.**Timeline**(***kwargs*)
Bases: gi.repository.Gtk.DrawingArea

__draw(*widget*, *context*)

begintime

do_get_preferred_height()
get_preferred_height(self) -> minimum_height:int, natural_height:int

do_get_property(*pspec*)

do_set_property(*pspec*, *value*)

endtime

indicatorsteptime

class parsley.gui.app_report.timelinepanel.**TimelinePanel**(***kwargs*)
Bases: gi.overrides.Gtk.Box

Initializer for a GObject based classes with support for property sets through the use of explicit keyword arguments.

__refresh(*_)

_populate(*dirpath*)

init_template(*self*)

pnltimelines = <gi._gtktemplate.Child object>

resizedetector = <gi._gtktemplate.Child object>

slctimeinterval = <gi._gtktemplate.Child object>

Module contents

The Parsley reporting gui.

class parsley.gui.app_report.**App**
Bases: *parsley.gui.gtk.volumeapp.AbstractVolumeApp*

_has_unsaved_changes()

_populate()

parsley.gui.app_report.**run**()

parsley.gui.aspect package

Submodules

parsley.gui.aspect.applypathacceptor module

parsley.gui.aspect.applypathacceptor.**changeguides**(*cfg*, *sync*)

parsley.gui.aspect.defaults module

parsley.gui.aspect.defaults.**changeguides**(*cfg*, *sync*)

parsley.gui.aspect.defaults.**changeguides_readonly**(*cfg*, *sync*)

parsley.gui.aspect.highlevelcustomization module

parsley.gui.aspect.highlevelcustomization.**changeguides**(*cfg*, *sync*)

parsley.gui.aspect.logging module

parsley.gui.aspect.logging.**changeguides**(*cfg*, *sync*)

parsley.gui.aspect.metadata module

parsley.gui.aspect.metadata.**changeguides**(*cfg*, *sync*)

parsley.gui.aspect.permissions module

parsley.gui.aspect.permissions.**changeguides**(*cfg*, *sync*)

parsley.gui.aspect.remove module

parsley.gui.aspect.remove.**changeguides**(*cfg*, *sync*)

parsley.gui.aspect.revisiontracking module

parsley.gui.aspect.revisiontracking.**changeguides**(*cfg*, *sync*)

Module contents

Gui helpers for aspects (see *parsley.aspect*).

parsley.gui.filesystem package

Submodules

parsley.gui.filesystem.local module

class parsley.gui.filesystem.local.**LocalFilesystemGuiCreateHelper**
Bases: object

configfs(*entry*, *sync*)

parsley.gui.filesystem.sshfs module

class parsley.gui.filesystem.sshfs.**SshFilesystemGuiCreateHelper**
Bases: object

configfs(*entry*, *sync*)

Module contents

Gui helpers for filesystems (see *parsley.filesystem*).

parsley.gui.gtk package

Subpackages

parsley.gui.gtk.volumeapp package

Module contents

A gui for manually resolving filesystem sync conflicts.

class parsley.gui.gtk.volumeapp.**AbstractVolumeApp**
Bases: object

__confirm_drop_unsaved(*onconfirmed*)

__handleclose(*_)

__opendir(*dirpath*)

__opendirdialog(*_)

_add_action_button(*button*)

abstract _has_unsaved_changes()

_populate()

_set_application_title(*title*)

_set_body(*widget*)

_set_welcome_text(*text*)

show()

Submodules

parsley.gui.gtk.userfeedback module

Helper class for user feedback dialogs.

class parsley.gui.gtk.userfeedback.**UserFeedbackController**(*owner*)
Bases: object

choicedialog(*question*, *choices*)
Shows a choice dialog to the user and returns the selected item (or None when cancelled).

Parameters

- **question** (*str*) – The question text to show.
- **choices** (*List[str]*) – The list of choices the user has to select from.

Return type Optional[int]

filesystemdialog(*fstype='file'*, *question=''*, *startpath=None*)
Shows a file/directory selection dialog to the user and returns the path to the selected item (or `None`).

Parameters

- **fstype** (*str*) – The type of filesystem items to select.
- **question** (*str*) – The question text to show.
- **startpath** (*Optional[str]*) – The directory path to start in.

Return type Optional[str]

inputdialog(*question*, *defaultanswer=''*)
Shows an input dialog to the user and returns the entered text (or `None` when cancelled).

Parameters

- **question** (*str*) – The question text to show.
- **defaultanswer** (*str*) – The default answer text that is written to the text field when showing.

Return type Optional[str]

messagedialog(*message*, *buttons=None*)
Shows a message dialog to the user and returns the index of the selected button.

Parameters

- **message** (*str*) – The message text to show.
- **buttons** (*Optional[List[str]]*) – The buttons to offer.

Return type int

multilineinputdialog(*question*, *defaultanswer=''*)
Like *inputdialog()* but multi-line capable.

Parameters

- **question** (*str*) – The question text to show.
- **defaultanswer** (*str*) – The default answer text that is written to the text field when showing.

Return type Optional[str]

Module contents

class parsley.gui.gtk.**EntryDialog**(*is_multiline=False*, ***kwargs*)
Bases: `gi.overrides.Gtk.MessageDialog`

Initializer for a GObject based classes with support for property sets through the use of explicit keyword arguments.

do_get_property(*pspec*)

do_set_property(*pspec*, *value*)

entry_text

run_and_get_entry_text()

class parsley.gui.gtk.**ParsleyGtkBuilder**
Bases: `gi.overrides.Gtk.Builder`

parsley.gui.gtk.**load_common_css**()

Return type None

parsley.gui.gtk.**load_css**(*path*)

Parameters **path** (*str*) –

Return type None

parsley.gui.gtk.**popover_menu**(*actions*, *align_to*)

Parameters

- **actions** (*List[Tuple[str, Callable[[], None]]]*) –
- **align_to** (*gi.overrides.Gtk.Widget*) –

Return type None

parsley.gui.icons package

Module contents

Submodules

parsley.gui.apidef module

Interface for publishing api information to the gui and for querying that infos.

class parsley.gui.apidef.**InvalidUserFeedbackController**
Bases: object

parsley.gui.apidef.**getfsname**(*fs*, *i*)
Returns a name for a filesystem (even if no 'name' param is set).

Parameters

- **fs** (parsley.config.config.ParsleyConfiguration.Filesystem) –
- **i** (*int*) –

Return type str

parsley.gui.apidef.**getnumstring**(*i*)
Returns a nice index representation, like '2nd' for 2.

Parameters **i** (*int*) –

Return type str

parsley.gui.apidef.**getregisteredchangeguides**()
Returns a list of all registered graphical change guides. See registerchangeguide.

parsley.gui.apidef.**getregisteredfilesystemhelpers**()
Returns a list of all registered filesystem implementations (together with gui helpers). See registerfilesystem.

parsley.gui.apidef.**registerchangeguide**()
Returns a function that registers a graphical change guide. A change guide is a graphical dialog (mostly with messageboxes, inputboxes, choiceboxes) that helps the user to configure a certain part of parsley functionality.

parsley.gui.apidef.**registerfilesystemhelper**(*clss*)
Returns a function that registers a filesystem implementation (together with gui helpers). This is mostly used for providing graphical aid in making a parsley configuration.

parsley.gui.helpers module

class parsley.gui.helpers.**ParsleyRunThread**(*listener*, *datadir*, *configfile*, *syncname*)
Bases: threading.Thread

This constructor should always be called with keyword arguments. Arguments are:

group should be None; reserved for future extension when a ThreadGroup class is implemented.

target is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.

name is the thread name. By default, a unique name is constructed of the form "Thread-N" where N is a small decimal number.

args is the argument tuple for the target invocation. Defaults to ().

kwargs is a dictionary of keyword arguments for the target invocation. Defaults to {}.

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread.__init__()) before doing anything else to the thread.

Parameters

- **listener** (parsley.gui.helpers.ParsleyRunThreadListener) –
- **datadir** (*str*) –
- **configfile** (*str*) –
- **syncname** (*str*) –

cancel()

Return type None

run()
Method representing the thread's activity.

You may override this method in a subclass. The standard run() method invokes the callable object passed to the object's constructor as the target argument, if any, with sequential and keyword arguments taken from the args and kwargs arguments, respectively.

class parsley.gui.helpers.**ParsleyRunThreadListener**
Bases: object

abstract ended(*returnvalue*)

Parameters **returnvalue** (*int*) –

Return type None

abstract output(*txt*)

Parameters **txt** (*str*) –

Return type None

parsley.gui.helpers.**_translator**()

Return type Callable[[str], str]

parsley.gui.helpers.**call_parsley_tool**(*toolname*, *, *syncname*, *datadir*)

Parameters

- **toolname** (*str*) –
- **syncname** (*str*) –
- **datadir** (*str*) –

Return type None

parsley.gui.helpers.**find_volume_rootpath**(*path*)

Parameters **path** (*str*) –

Return type Optional[str]

parsley.gui.helpers.**get_volume_paths_for_sync**(*syncname*, ***, *datadir*)

Parameters

- **syncname** (*str*) –
- **datadir** (*str*) –

Return type List[str]

parsley.gui.helpers.**openconfig**(*cfgfile*)

Parameters **cfgfile** (*str*) –

Return type Optional[*parsley.config.config.ParsleyConfiguration*]

parsley.gui.helpers.**parsley_cmdline**(**params*, *toolname='parsley'*)

Parameters

- **params** (*str*) –
- **toolname** (*str*) –

Return type List[str]

parsley.gui.helpers.**saveconfig**(*cfg*)

Parameters **cfg** (parsley.config.config.ParsleyConfiguration) –

Return type None

parsley.gui.report module

class parsley.gui.report.**LogMessageWithTime**(*logmessage*, *time*)

Bases: object

Parameters

- **logmessage** (parsley.logger.logger.LogMessage) –
- **time** (*datetime.datetime*) –

class parsley.gui.report.**PerformanceData**(*syncruns*, *eventhandlers*, *filesystems*, *timingdata*)

Bases: object

Parameters

- **syncruns** (*List[str]*) –
- **eventhandlers** (*List[str]*) –

- **filesystems** (*List[str]*) –
- **timingdata** (*Dict[str, PerformanceData.Sync]*) –

class EventHandler(*name*, *eventname*)
Bases: object

Parameters

- **name** (*str*) –
- **eventname** (*str*) –

class Filesystem(*name*)
Bases: object

Parameters **name** (*str*) –

class Loader
Bases: object

classmethod __read_events(*_levents*, *_eventlist*, *_eventhandlerset*, *_filesystemset*)

add_data_available_changed_handler(*fct*)

add_query_result_available_changed_handler(*fct*)

property data_available
Return type bool

get_data()
Return type *parsley.gui.report.PerformanceData*

get_query_result()
Return type *parsley.gui.report.PerformanceData.QueryResult*

query(*, *horizontally*, *vertically*, *aggregation*, *sync=None*, *syncrun=None*, *eventhandler=None*, *filesystem=None*, *path=None*)
Parameters
- **horizontally** (*str*) –
- **vertically** (*str*) –
- **aggregation** (*str*) –
- **sync** (*Optional[str]*) –
- **syncrun** (*Optional[str]*) –
- **eventhandler** (*Optional[str]*) –
- **filesystem** (*Optional[str]*) –
- **path** (*Optional[str]*) –

property query_result_available
Return type bool

set_path(*path*)
Parameters **path** (*str*) –
Return type None

class Path(*path*)
Bases: object

Parameters **path** (*str*) –

class QueryResult(*row_names*, *column_names*, *result*)
Bases: object

Parameters

- **row_names** (*List[str]*) –
- **column_names** (*List[str]*) –
- **result** (*List[List[float]]*) –

class Sync(*name*)
Bases: object

Parameters **name** (*str*) –

class SyncRun(*name*)
Bases: object

Parameters **name** (*str*) –

class Table
Bases: object

add(*rowname*, *colname*, *value*)
Parameters
- **rowname** (*str*) –
- **colname** (*str*) –
- **value** (*float*) –

Return type None

property eventhandlers

Return type t.List[str]

property filesystems

Return type t.List[str]

property syncruns

Return type t.List[str]

property synctasks

Return type t.List[str]

property timingdata

Return type t.Dict[str, 'PerformanceData.Sync']

class parsley.gui.report.**SyncRunLog**(*sync*, *syncrun*, *messages*, *begintime*, *endtime*, *crashed*)
Bases: object

Parameters

- **sync** (*str*) –
- **syncrun** (*str*) –
- **messages** (*List[parsley.gui.report.LogMessageWithTime]*) –
- **begintime** (*datetime.datetime*) –
- **endtime** (*datetime.datetime*) –
- **crashed** (*bool*) –

parsley.gui.report.**try_load_log_from_syncdir**(*path*, *sync='?'*)

Parameters

- **path** (*str*) –

- **sync** (*str*) –

Return type List[*parsley.gui.report.SyncRunLog*]

Module contents

Graphical user interfaces.

parsley.logger package

Subpackages

parsley.logger.formatter package

Submodules

parsley.logger.formatter.abstractlogformat module

Abstract base class for log formatters.

class parsley.logger.formatter.abstractlogformat.**Logformat**

Bases: object

Abstract base class for log message formatters (html, plain, ...). For existing implementations, see parsley.logger.formatter.

footer()

Returns a footer string for the log output (like an html footer).

Return type str

header()

Returns a header string for the log output (like an html header).

Return type str

log(*logmessage*)

Renders the information for one log message into a formatted string (plaintext, html, ...).

Parameters **logmessage** (parsley.logger.logger.LogMessage) – The parsley.logger.logger.LogMessage log message.

Return type str

parsley.logger.formatter.htmllogformat module

class parsley.logger.formatter.htmllogformat.**HtmlLogformat**

Bases: *parsley.logger.formatter.abstractlogformat.Logformat*

footer()

Returns a footer string for the log output (like an html footer).

static get_htmlcolor_for_severity(*severity*)

header()

Returns a header string for the log output (like an html header).

log(*logmessage*)
Renders the information for one log message into a formatted string (plaintext, html, ...).

Parameters **logmessage** – The parsley.logger.logger.LogMessage log message.

parsley.logger.formatter.plaintextlogformat module

class parsley.logger.formatter.plaintextlogformat.**PlaintextLogformat**(*maxlen=80*, *color=None*)
Bases: *parsley.logger.formatter.abstractlogformat.Logformat*

footer()
Returns a footer string for the log output (like an html footer).

header()
Returns a header string for the log output (like an html header).

log(*logmessage*)
Renders the information for one log message into a formatted string (plaintext, html, ...).

Parameters **logmessage** – The parsley.logger.logger.LogMessage log message.

Module contents

Transforming log message structures to strings.

parsley.logger.loggerout package

Submodules

parsley.logger.loggerout.abstractloggerout module

Abstract base class for a logger output.

class parsley.logger.loggerout.abstractloggerout.**Loggerout**
Bases: object

Abstract base class for log message sinks. For existing implementations, see parsley.logger.loggerout.

flush(*wasused*)
Commits the log content to the sink and closes the logger.

Parameters **wasused** (*bool*) – If there was any useful logged message (instead of just headers and footers).

Return type None

log(*content*)
Writes the rendered content to the logger (can be log messages, html header or footer, ...).

Parameters **content** (*str*) –

Return type None

parsley.logger.loggerout.externalprogramloggerout module

class parsley.logger.loggerout.externalprogramloggerout.**ExternalProgramLoggerout**(***kwa*)
Bases: *parsley.logger.loggerout.abstractloggerout.Loggerout*

flush(*wasused*)
Commits the log content to the sink and closes the logger.

Parameters **wasused** – If there was any useful logged message (instead of just headers and footers).

log(*content*)
Writes the rendered content to the logger (can be log messages, html header or footer, ...).

parsley.logger.loggerout.filestreamloggerout module

class parsley.logger.loggerout.filestreamloggerout.**FilestreamLoggerout**(*filename=None*)
Bases: *parsley.logger.loggerout.abstractloggerout.Loggerout*

flush(*wasused*)
Commits the log content to the sink and closes the logger.

Parameters **wasused** – If there was any useful logged message (instead of just headers and footers).

log(*content*)
Writes the rendered content to the logger (can be log messages, html header or footer, ...).

Module contents

Output destinations for log message strings.

Submodules

parsley.logger.logger module

Logging messages.

class parsley.logger.logger.**LogMessage**(*sync*, *subject*, *verb*, *comment*, *severity*, *symbol*)
Bases: object

A single log message.

This is used in the logging backend. You do not need this class just for logging.

Parameters

- **sync** (*str*) –
- **subject** (*str*) –
- **verb** (*str*) –
- **comment** (*str*) –
- **severity** (*int*) –
- **symbol** (*str*) –

class parsley.logger.logger.**Logger**(*formatter*, *loggerout*, *minseverity=3*, *maxseverity=None*, *enabled=True*)

Bases: object

Used by higher layers for logging messages. Contains a parsley.logger.formatter.abstractlogformat.Logformat and a parsley.logger.loggerout.abstractloggerout.Loggerout and defines which severity span should actually be logged.

Parameters

- **formatter** (parsley.logger.formatter.abstractlogformat.Logformat) –
- **loggerout** (parsley.logger.loggerout.abstractloggerout.Loggerout) –
- **minseverity** (*Optional[int]*) –
- **maxseverity** (*Optional[int]*) –
- **enabled** (*bool*) –

flush()

Closes the logger and commits the content.

Return type None

log(*sync*, *subject*, *verb*, *comment*, *severity*, *symbol*)

Logs a message.

Parameters

- **sync** (*str*) – The name of the synchronization task that is presented as the source (arbitrary string).
- **subject** (*str*) – The subject string.
- **verb** (*str*) – The verb string.
- **comment** (*str*) – The comment string.
- **severity** (*int*) – The severity (parsley.logger.logger.Severity).
- **symbol** (*str*) – The event symbol string.

Return type None

class parsley.logger.logger.**Severity**

Bases: object

Enumeration for the severity of log messages.

DEBUG = 2

DEBUGVERBOSE = 1

ERROR = 6

IMPORTANT = 4

INFO = 3

MOREIMPORTANT = 5

_MAX_TYPICAL = 6

This is the maximum severity value for typical log messages. Everything above is for special purpose categories.

Module contents

Logging messages. See *parsley.logger.logger*.

parsley.preparation package

Submodules

parsley.preparation.abstractpreparation module

Implement custom preparations by subclassing *Preparation*.

class parsley.preparation.abstractpreparation.**Preparation**
Bases: object

Abstract base class for all logic that does preparation work for the actual synchronization run in advance or afterwards (like mounting filesystems, cleaning up stuff).

disable(*runtime*)
Undo the preparation after the synchronization (like unmounting).

Parameters **runtime** (parsley.runtime.runtime.RuntimeData) –

Return type None

enable(*runtime*)
Runs the preparation work.

Parameters **runtime** (parsley.runtime.runtime.RuntimeData) –

Return type None

ensuredisabledafter()
Defines if this preparation is required to be disabled after the synchronization ended. For example, it is an error situation if a certain mountpoint is mounted even after this preparation has unmounted it.

Return type bool

ensuredisabledbefore()
Defines if this preparation is required to be disabled before the synchronization begins. For example, it is an error situation if a certain mountpoint is already mounted before this preparation has mounted it.

Return type bool

ensureenabled()
Defines if this preparation is required to be successful for the ongoing process of synchronization. For example, a preparation that just does some uncritical cleanup, should not stop the complete synchronization if it fails.

Return type bool

getstate(*runtime*)
Checks if the preparation is successfully done or not. The parsley engine will check the state with this method at certain times and decide what to do depending on the state and the return values of ensuredisabledbefore, ensuredisabledafter and ensureenabled.

Parameters **runtime** (parsley.runtime.runtime.RuntimeData) –

Return type bool

parsley.preparation.mountpreparation module

class parsley.preparation.mountpreparation.**MountPreparation**(*, *src*, *tgt*, ***kwargs*)

Bases: *parsley.preparation.abstractpreparation.Preparation*

Mounts filesystems to a mountpoint.

Parameters

- **src** (*str*) –
- **tgt** (*str*) –

static _checkmountpointempty(*tgt*)

disable(*runtime*)
Undo the preparation after the synchronization (like unmounting).

enable(*runtime*)
Runs the preparation work.

getstate(*runtime*)
Checks if the preparation is successfully done or not. The parsley engine will check the state with this method at certain times and decide what to do depending on the state and the return values of ensuredisabledbefore, ensuredisabledafter and ensureenabled.

exception parsley.preparation.mountpreparation.**MountPreparationError**
Bases: *parsley.exceptions.ParsleyError*

exception parsley.preparation.mountpreparation.**MountpointNotEmptyError**(*mountpoint*)
Bases: *parsley.preparation.mountpreparation.MountPreparationError*

Parameters **mountpoint** (*str*) –

exception parsley.preparation.mountpreparation.**UnableToGetMountListError**(*msg*)
Bases: *parsley.preparation.mountpreparation.MountPreparationError*

Parameters **msg** (*str*) –

exception parsley.preparation.mountpreparation.**UnableToMountError**(*mountpoint*, *msg*)
Bases: *parsley.preparation.mountpreparation.MountPreparationError*

Parameters

- **mountpoint** (*str*) –
- **msg** (*str*) –

exception parsley.preparation.mountpreparation.**UnableToUnmountError**(*mountpoint*, *msg*)
Bases: *parsley.preparation.mountpreparation.MountPreparationError*

Parameters

- **mountpoint** (*str*) –
- **msg** (*str*) –

parsley.preparation.preparator module

Helper for enabling and disabling preparations.

class parsley.preparation.preparator.**Preparator**(*runtime*, *preparation*)
Bases: object

Parameters

- **runtime** (parsley.runtime.runtime.RuntimeData) –
- **preparation** (parsley.preparation.abstractpreparation.Preparation) –

static _logandthrowexception(*runtime*, *exclass*, *olde=None*)

Parameters

- **runtime** (parsley.runtime.runtime.RuntimeData) –
- **exclass** (*Type[Exception]*) –
- **olde** (*Optional[Exception]*) –

Return type None

parsley.preparation.sshfsmountpreparation module

class parsley.preparation.sshfsmountpreparation.**SshfsMountPreparation**(*, *src*, *tgt*, *idfile=None*, *password=None*, *port=22*, *timeout='10s'*, ***kwargs*)
Bases: *parsley.preparation.mountpreparation.MountPreparation*

Mounts remote filesystems with sshfs.

Parameters

- **src** (*str*) –
- **tgt** (*str*) –
- **idfile** (*Optional[str]*) –
- **password** (*Optional[str]*) –
- **port** (*int*) –
- **timeout** (*Union[datetime.timedelta, str]*) –

disable(*runtime*)
Undo the preparation after the synchronization (like unmounting).

enable(*runtime*)
Runs the preparation work.

static translate_to_alternative_endpoint(*src*, *port*, *runtime*)

Parameters

- **src** (*str*) –
- **port** (*int*) –
- **runtime** (parsley.runtime.runtime.RuntimeData) –

Return type Tuple[str, int]

Module contents

Pluggable initialization/preparation steps that can be wrapped around task executions by adding them to the task configuration. Implement custom preparations by subclassing :py:class:`parsley.preparation.abstractpreparation.Preparation´.

parsley.runtime package

Submodules

parsley.runtime.commonimports module

Imports everything that is typically used by the end user in a Parsley config file.

parsley.runtime.datastorage module

Implements an abstraction layer for storing arbitrary process data, like sync states or file history information.

class parsley.runtime.datastorage.**StorageDepartment**

Bases: object

One main hive for data storage.

A main hive has a name and can store data in a tree-like structure (as in filesystems). Different subclasses exist and can be configured in different flavours.

VALUE_FILENAME = '.~parsley+~value.'

Internally used as special file name for value storage.

get_filesystem(*, *filesystem*, *sync=None*)

Gets the hive filesystem for more complex custom operations.

Parameters

- **filesystem** (*Optional[parsley.filesystem.abstractfilesystem.Filesystem]*) – The filesystem to get the data for. Not used in some implementations.
- **sync** (*Optional[parsley.syncengine.sync.Sync]*) – The sync. Typically not needed to specify it.

Return type *parsley.filesystem.abstractfilesystem.Filesystem*

get_value_bytes(*, *path='/'*, *filesystem*, *sync=None*, *throwonnotexists=True*, *defaultval=b''*)

Gets a value as *bytes* (byte array).

Parameters

- **path** (*str*) – The data path.

- **filesystem** (*Optional[parsley.filesystem.abstractfilesystem.Filesystem]*) – The filesystem to get the data for. Not used in some implementations.
- **sync** (*Optional[parsley.syncengine.sync.Sync]*) – The sync. Typically not needed to specify it.
- **throwonnotexists** (*bool*) – If to throw an exception if there is no data stored for this path.
- **defaultval** (*Optional[bytes]*) – The default value (when no one exists and throwonnotexists=False).

Return type Optional[bytes]

get_value_string(*, *path='/'*, *filesystem=None*, *sync=None*, *throwonnotexists=True*, *defaultval=''*)

Gets a value as *string*.

Parameters

- **path** (*str*) – The data path.
- **filesystem** (*Optional[parsley.filesystem.abstractfilesystem.Filesystem]*) – The filesystem to get the data for. Not used in some implementations.
- **sync** (*Optional[parsley.syncengine.sync.Sync]*) – The sync. Typically not needed to specify it.
- **throwonnotexists** (*bool*) – If to throw an exception if there is no data stored for this path.
- **defaultval** (*Optional[str]*) – The default value (when no one exists and throwonnotexists=False).

Return type Optional[str]

has_value(*, *path='/'*, *filesystem*, *sync=None*)

Checks if some value is stored for a path.

Parameters

- **path** (*str*) – The data path.
- **filesystem** (*Optional[parsley.filesystem.abstractfilesystem.Filesystem]*) – The filesystem to check. Not used in some implementations.
- **sync** (*Optional[parsley.syncengine.sync.Sync]*) – The sync. Typically not needed to specify it.

Return type bool

list_path(*, *path*, *filesystem*, *sync=None*)

Returns a list of names of all subitems in the given path. Those can either lead to further subitems, or can have a value, or both!

Parameters

- **path** (*str*) – The data path.
- **filesystem** (*Optional[parsley.filesystem.abstractfilesystem.Filesystem]*) – The filesystem to list the items for. Not used in some implementations.
- **sync** (*Optional[parsley.syncengine.sync.Sync]*) – The sync. Typically not needed to specify it.

Return type List[str]

remove_value(*, *path*, *filesystem*, *sync=None*, *recursive=False*, *throwonnotexists=True*)
Removes a value from storage.

Parameters

- **path** (*str*) – The data path.
- **filesystem** (*Optional[parsley.filesystem.abstractfilesystem.Filesystem]*) – The filesystem to remove data from. Not used in some implementations.
- **sync** (*Optional[parsley.syncengine.sync.Sync]*) – The sync. Typically not needed to specify it.
- **recursive** (*bool*) – If to remove also all subitems.
- **throwonnotexists** (*bool*) – If to throw an exception if there is no data stored for this path.

Return type None

set_value(*value*, *, *path='/'*, *filesystem*, *sync=None*)
Sets and stores a value.

Parameters

- **value** (*AnyStr*) – The value data.
- **path** (*str*) – The data path.
- **filesystem** (*Optional[parsley.filesystem.abstractfilesystem.Filesystem]*) – The filesystem to store the data for. Not used in some implementations.
- **sync** (*Optional[parsley.syncengine.sync.Sync]*) – The sync. Typically not needed to specify it.

Return type None

class parsley.runtime.datastorage.**StorageLocation**
Bases: object

Enumeration of storage locations.

SYNC_VOLUME = 'sync_volume'

SYSTEM = 'system'

class parsley.runtime.datastorage.**StorageScope**
Bases: object

Enumeration of storage scopes.

PER_SYNC = 'per_sync'

SHARED = 'shared'

class parsley.runtime.datastorage.**SyncVolumeStorageDepartment**(*runtime*, *name*, *scope*)
Bases: *parsley.runtime.datastorage.StorageDepartment*

Implementation of StorageDepartment for storage of data inside the sync volume (i.e. potentially on the remote device).

@note Although the data is stored in the sync volume, this data is not part of what gets synchronized
(but resides in a hidden part).

_get_rootpath(*sync*)

get_filesystem(*, *filesystem*, *sync=None*)
Gets the hive filesystem for more complex custom operations.

Parameters

- **filesystem** – The filesystem to get the data for. Not used in some implementations.
- **sync** – The sync. Typically not needed to specify it.

get_value_bytes(*, *path='/'*, *filesystem*, *sync=None*, *throwonnotexists=True*, *defaultval=b''*)
Gets a value as *bytes* (byte array).

Parameters

- **path** – The data path.
- **filesystem** – The filesystem to get the data for. Not used in some implementations.
- **sync** – The sync. Typically not needed to specify it.
- **throwonnotexists** – If to throw an exception if there is no data stored for this path.
- **defaultval** – The default value (when no one exists and throwonnotexists=False).

get_value_path(*sync*, *path='/'*)

set_value(*value*, *, *path='/'*, *filesystem=''*, *sync=None*)
Sets and stores a value.

Parameters

- **value** – The value data.
- **path** – The data path.
- **filesystem** – The filesystem to store the data for. Not used in some implementations.
- **sync** – The sync. Typically not needed to specify it.

class parsley.runtime.datastorage.**SystemStorageDepartment**(*runtime*, *name*, *scope*)
Bases: *parsley.runtime.datastorage.StorageDepartment*

Implementation of StorageDepartment for storage of data inside the system, i.e. not on some remote place.

_get_rootpath(*sync*)

get_filesystem(*, *filesystem=None*, *sync=None*)
Gets the hive filesystem for more complex custom operations.

Parameters

- **filesystem** – The filesystem to get the data for. Not used in some implementations.
- **sync** – The sync. Typically not needed to specify it.

get_value_bytes(*, *path='/'*, *filesystem=None*, *sync=None*, *throwonnotexists=True*, *defaultval=b''*)
Gets a value as *bytes* (byte array).

Parameters

- **path** – The data path.
- **filesystem** – The filesystem to get the data for. Not used in some implementations.
- **sync** – The sync. Typically not needed to specify it.
- **throwonnotexists** – If to throw an exception if there is no data stored for this path.
- **defaultval** – The default value (when no one exists and throwonnotexists=False).

get_value_path(*sync*, *path='/'*)

has_value(***kwa*)
Checks if some value is stored for a path.

Parameters

- **path** – The data path.
- **filesystem** – The filesystem to check. Not used in some implementations.
- **sync** – The sync. Typically not needed to specify it.

set_value(*value*, ***, *path='/'*, *filesystem=None*, *sync=None*)
Sets and stores a value.

Parameters

- **value** – The value data.
- **path** – The data path.
- **filesystem** – The filesystem to store the data for. Not used in some implementations.
- **sync** – The sync. Typically not needed to specify it.

parsley.runtime.datastorage.**get_storage_department**(*ctx*, *name*, ***, *location='sync_volume'*, *scope='per_sync'*)
Creates a StorageDepartment for storage of process data.

Parameters

- **ctx** (parsley.runtime.runtime.RuntimeData) – The current runtime context.
- **name** (*str*) – The name of this storage department name.
- **location** (*str*) – The storage location. One of StorageLocation.
- **scope** (*str*) – The storage scope. One of StorageScope.

Return type Union[*parsley.runtime.datastorage.SyncVolumeStorageDepartment*, *parsley.runtime.datastorage.SystemStorageDepartment*]

parsley.runtime.projectinformations module

parsley.runtime.returnvalue module

Parsley engine return values.

class parsley.runtime.returnvalue.**ReturnValue**
Bases: object

Flags that roughly describe the success of a synchronization run. The return value you get on command line is composed from this flags.

DIRTY = 32
Something remained dirty and needs a new synchronization run soon.

ERROR = 1
An error occurred.

ERROR_EXECUTION = 9
An error occurred while executing the synchronization itself (excluding preparations, . . .).

ERROR_INITIALIZATION = 3
An error occurred in the initialization of the synchronization itself.

ERROR_PREPARATION = 5
An error occurred while enabling one of the specified parsley.preparation.abstractpreparation.Preparation before execution.

ERROR_UNPREPARATION = 17
An error occurred while disabling one of the specified parsley.preparation.abstractpreparation.Preparation after execution.

SUCCESS = 0
Everything went perfectly okay.

parsley.runtime.runtime module

The implementation for some Parsley core features.

class parsley.runtime.runtime.**RuntimeData**(*datadir*, *loggers*, *syncs*)
Bases: object

Runtime data for synchronization runs used by the parsley engine and some higher layers. Used by the synchronization implementations for logging, for communicating success values and for retrieving/setting some environment values. There is one instance for each parsley engine run (which can do multiple synchronizations).

Parameters

- **datadir** (*str*) – Path to the directory for local parsley control data.
- **loggers** (*List[parsley.logger.logger.Logger]*) – List of loggers.
- **syncs** (*List[parsley.syncengine.sync.Sync]*) – List of sync tasks.

clone(*merge_also_from=None*, *merge_also=None*)
Clones this object.

This is used for providing objects for repeating sub-processes, so they always get a fresh but pre-populated RuntimeData object. This method returns a shallow copy, which is used for writing stuff to the original from a clone in some ways.

Parameters

- **merge_also_from** (*Optional[Any]*) – Take all members from this object into the new clone as well.
- **merge_also** (*Optional[Dict[str, Optional[Any]]]*) – Take all entries from this dictionary as members into the new clone as well.

Return type *parsley.runtime.runtime.RuntimeData*

get_retval()
Gets the return value as combination of parsley.runtime.returnvalue.ReturnValue.

Return type int

log(*, *subject=''*, *verb=''*, *comment=''*, *severity=3*, *symbol=''*)
This is the log method that is actually used by clients.

Parameters

- **subject** (*str*) – The name of the subject, e.g. a file name (arbitrary string).

- **verb** (*str*) – A description what happened with the subject, e.g. 'deleted' or 'created' (arbitrary string).
- **comment** (*str*) – A description following after the verb (arbitrary string).
- **severity** (*int*) – A parsley.logger.severity.Severity describes how important this message is. This value decides if the message will be really logged.
- **symbol** (*str*) – One character that describes the kind of log message (arbitrary string).

Return type None

mark_dirty()
Marks the current sync run as dirty, which leads to a faster re-execution (e.g. since one or more file was skipped).

Return type None

set_retval(*flag*)
Sets a flag from parsley.runtime.returnvalue.ReturnValue to the return value.

Parameters **flag** (*int*) –

Return type None

parsley.runtime.successtracker module

Tracks the success of sync runs, invokes warning messages after some time of sync problems, ...

class parsley.runtime.successtracker.**SuccessTracker**(*runtime*)
Bases: object

Tracks the success of sync runs.

Used internally for logging warnings when a synchronization did not run successfully since a certain timespan. It stores timestamps on the filesystem for this functionality.

Parameters **runtime** (parsley.runtime.runtime.RuntimeData) –

begincall(*sync*)
Called by the infrastructure when a sync run begins, for bookkeeping.

Parameters **sync** (parsley.syncengine.sync.Sync) –

Return type None

end()
Signals that a task execution was scheduled for now and the process is over. This happens for successful and for failed executions and triggers a warning notification in certain situations.

Return type None

forceexecution(*sync*)
Marks a sync task so it gets executed next time regardless of the sync interval.

Parameters **sync** (parsley.syncengine.sync.Sync) –

Return type None

getlastsuccessfulcall(*sync*)
Gets a tuple of the time of last successful execution and the forced flag of a task.

Parameters **sync** (parsley.syncengine.sync.Sync) –

Return type Tuple[Optional[datetime.datetime], bool]

setscope(*syncs*)
Sets the list of sync tasks for processing.

Parameters **syncs** (*List[parsley.syncengine.sync.Sync]*) –

Return type None

shall_skip(*sync*)
Determines if a sync task should be skipped now.

Parameters **sync** (parsley.syncengine.sync.Sync) –

Return type bool

successfulcall(*sync*)
Marks a sync task as successfully executed.

Parameters **sync** (parsley.syncengine.sync.Sync) –

Return type None

Module contents

Parsley core parts.

parsley.syncengine package

Submodules

parsley.syncengine.benchmark module

Time measurement of sync runs.

class parsley.syncengine.benchmark.**BenchmarkRun**(*syncruntime*)
Bases: object

Collects benchmark data for a sync run.

This collection is populated with data by the engine during execution and stored afterwards, so external tools can inspect this data and get detailed ideas about the runtime performance.

Parameters **syncruntime** (parsley.runtime.runtime.RuntimeData) –

beginsync()

Return type None

beginsyncevent(*syncevent*, *path*)

Parameters

- **syncevent** (*str*) –
- **path** (*str*) –

Return type None

beginsynceventhandler(*handler*, *filesystem*)

Parameters

- **handler** (parsley.syncengine.sync.Sync.EventHandler) –

- **filesystem** (parsley.filesystem.abstractfilesystem.Filesystem) –

Return type None

endsync()

Return type None

endsyncevent()

Return type None

endsynceventhandler()

Return type None

get_report()

version = 1

parsley.syncengine.common module

Boring common data structures.

class parsley.syncengine.common.**EntryType**
Bases: object

A type of an entry (sometimes the word 'file' is wrongly used instead of 'entry') in the filesystem.

Directory = 'D'
Directory

File = 'F'
Regular file

Link = 'L'
Symlink

parsley.syncengine.common.**SKIP = <object object>**
A flag that means skipping parts of execution. It is a general purpose signalling object. It might be interpreted from different parties at different places in a different way. Used e.g. in master election as a special master for skipping the synchronization process (read parsley.syncengine.syncruntime.SyncEventRuntime.skip_master_filesystem_promotion) and in the update process for signalling skipping due to lately found conflicts (read parsley.aspect.execute_only_if_not_update_set_skipped).

exception parsley.syncengine.common.**SyncConfigurationError**
Bases: *parsley.syncengine.common.SyncError*

Errors while processing the sync configuration.

exception parsley.syncengine.common.**SyncError**
Bases: *parsley.exceptions.ParsleyError*

Errors happening in the sync engine.

class parsley.syncengine.common.**SyncEvent**
Bases: object

Names of well known synchronization events.

BeginSync = 'BeginSync'
Occurs once at the very beginning of a synchronization task execution.

CloseSync = 'CloseSync'
Occurs after EndSync, even when a fatal error occurred, i.e. typically an unhandled exception. This event is not used for typical stuff but only for infrastructures.

EndSync = 'EndSync'
Occurs once at the very end of a synchronization task execution.

LogCreate = 'LogCreate'
Logging.

LogProblem = 'LogProblem'
Logging.

LogRemove = 'LogRemove'
Logging.

LogUpdate = 'LogUpdate'
Logging.

UpdateDir_AfterUpdate = 'UpdateDir_AfterUpdate'
Occurs in updating directories, when the directory was updated completely. Use this state for executing code after a directory was synchronized, e.g. for bookkeeping.

UpdateDir_ListDir = 'UpdateDir_ListDir'
Occurs in updating directories, when the list of entry names must be fetched. Use this state for listing the directory. The code must call *ctx.entrylist.update* with a list of entry names in order to participate to listing the directory.

UpdateDir_Prepare = 'UpdateDir_Prepare'
Occurs in updating directories, when the preparation phase takes places, but before the directory is listed. Use this state for preparing directory synchronization or for some bookkeeping.

UpdateItem_AfterUpdate = 'UpdateItem_AfterUpdate'
Occurs in updating items, when the actual update phase is done. Use this state for executing code after the update took place, e.g. for some bookkeeping.

UpdateItem_BeforeElectMaster = 'UpdateItem_BeforeElectMaster'
Occurs in updating items, when very early initialization can happen, before the actual master election phase begins. Use this state for executing code at the very beginning of item synchronization, even before the master election takes place, e.g. for some bookkeeping.

UpdateItem_CheckConflicts = 'UpdateItem_CheckConflicts'
Occurs in updating items, when a master filesystem is just elected, for checking for conflicts. Hook methods here which checks for conflicts between the master filesystem and the filesystems it is hooked for. Signal conflicts by calling *ctx.add_conflict*.

UpdateItem_ElectMaster = 'UpdateItem_ElectMaster'
Occurs in updating items, when the master election phase began. Elect a master filesystem here in order to control which filesystems gets updated from which one. Use parsley.syncengine.syncruntime.SyncEventRuntime.promote_master_filesystem on *ctx* in order to elect a master filesystem or parsley.syncengine.syncruntime.SyncEventRuntime.skip_master_filesystem_promotion for ultimatively vote for skipping the synchronization of this entry completely. In all subsequent steps, the elected master filesystem is available in *ctx.masterfs*.

UpdateItem_ResolveConflicts = 'UpdateItem_ResolveConflicts'
Occurs in updating items, when conflicts occurred, in order to solve them. If this phase cannot resolve the conflicts, the usual update is skipped. Use this state for executing code that can resolve conflicts detected in SyncEvent.UpdateItem_CheckConflicts. For signalling that a conflict is resolved, call *ctx.remove_conflict*. It may decide to set a new *ctx.masterfs* for a new master filesystem.

UpdateItem_SkippedDueConflicts = 'UpdateItem_SkippedDueConflicts'
Occurs in updating items, when conflicts were not resolved and the item is skipped due to that. Use this state for executing code that handles unresolved conflicts that led to skipping this item.

UpdateItem_Update_ExistsInMaster = 'UpdateItem_Update_ExistsInMaster'
Occurs in updating items, when the actual update must take place and there exists an item in the master filesystem (typical update scenario). Use this state for actually transfer content from the master to the filesystem on which the hook runs.

UpdateItem_Update_NotExistsInMaster = 'UpdateItem_Update_NotExistsInMaster'
Occurs in updating items, when the actual update must take place and there does not exist an item in the master filesystem (typical removal scenario). Use this state for actually remove content from the filesystem on which the hook runs.

UpdateItem_Update_Prepare = 'UpdateItem_Update_Prepare'
Occurs after conflict resolution took place and the actual update phase is just before beginning.

exception parsley.syncengine.common.**SyncExecutionError**
Bases: *parsley.syncengine.common.SyncError*

Errors while executing the synchronization task.

parsley.syncengine.core module

The engine which abstractly executes arbitrary synchronization tasks.

class parsley.syncengine.core.**CmdLine**(*args*)
Bases: object

Parser for command line arguments.

Parameters **args** (*List[str]*) –

configfile_or_default()

Return type str

datadir_or_default()

Return type str

class parsley.syncengine.core.**Core**
Bases: object

The Parsley engine. It is the lowest (or higest, if you want) layer, which coordinates all the synchronization work.

class CoreRuntime(*sync*)
Bases: object

Parameters **sync** (parsley.syncengine.sync.Sync) –

class _Myclass
Bases: object

static execute(*syncs*, *datadir*, *syncname=None*, *logging=None*, *throwexceptions=False*)
Executes the selected synchronization tasks.

Parameters

- **syncs** (*List[parsley.syncengine.sync.Sync]*) – a list of synchronization tasks existing in your configuration.

- **datadir** (*str*) – which directory to use for persistent control data (file lists, ...).
- **syncname** (*Optional[str]*) – which sync task shall be executed (None: all).
- **logging** (*Optional[List[parsley.logger.Logger]]*) – list of loggers of logging output.
- **throwexceptions** (*bool*) – If exceptions should be raised in error case.

Return type int

parsley.syncengine.core.**_createemptyconfigifneeded**(*cmdline*)

Parameters **cmdline** (parsley.syncengine.core.CmdLine) –

Return type None

parsley.syncengine.core.**main**(*cmdline=None*, *callexit=True*, *environment=None*)

Main method for execution from command line.

Parameters

- **cmdline** (*Optional[List[str]]*) –
- **callexit** (*bool*) –
- **environment** (*Optional[Dict[str, str]]*) –

Return type int

parsley.syncengine.entrylist module

Internal in-memory list of entries for one filesystem with (de-)serialization features.

class parsley.syncengine.entrylist.**CombinedEntryList**(*first*, *second*)

Bases: object

Parameters

- **first** (parsley.syncengine.entrylist.EntryList) –
- **second** (parsley.syncengine.entrylist.EntryList) –

addtag(*path*, *tag*)

exists(*path*)

foundfile(*path*, *ftype*, *size*, *mtime*, *param*, *last_synced_mtime=None*)

getftype(*path*)

getlastsyncedmtime(*path*)

getmtime(*path*)

getparam(*path*)

getsize(*path*)

gettags(*path*)

notfoundfile(*path*)

removefile(*path*)

setlastsyncedmtime(*path*, *mtime*)

updatefile(*path*, *ftype*, *size*, *mtime*, *param*)

class parsley.syncengine.entrylist.**EntryList**
Bases: object

addtag(*path*, *tag*)

Parameters

- **path** (*str*) –
- **tag** (*str*) –

Return type None

exists(*path*)

Parameters **path** (*str*) –

Return type bool

foundfile(*path*, *ftype*, *size*, *mtime*, *param*, *last_synced_mtime=None*)

Parameters

- **path** (*str*) –
- **ftype** (*str*) –
- **size** (*int*) –
- **mtime** (*datetime.datetime*) –
- **param** (*Optional[str]*) –
- **last_synced_mtime** (*Optional[datetime.datetime]*) –

Return type None

getftype(*path*)

Parameters **path** (*str*) –

Return type str

getlastsyncedmtime(*path*)

Parameters **path** (*str*) –

Return type Optional[datetime.datetime]

getmtime(*path*)

Parameters **path** (*str*) –

Return type datetime.datetime

getparam(*path*)

Parameters **path** (*str*) –

Return type Optional[str]

getsize(*path*)

Parameters **path** (*str*) –

Return type int

gettags(*path*)

Parameters **path** (*str*) –

Return type List[str]

notfoundfile(*path*)

Parameters **path** (*str*) –

Return type None

read(*cnt*)

Parameters **cnt** (*bytes*) –

Return type None

removefile(*path*)

Parameters **path** (*str*) –

Return type None

save()

Return type bytes

setlastsyncedmtime(*path*, *mtime*)

Parameters

- **path** (*str*) –
- **mtime** (*datetime.datetime*) –

Return type None

updatefile(*path*, *ftype*, *size*, *mtime*, *param*)

Parameters

- **path** (*str*) –
- **ftype** (*str*) –
- **size** (*int*) –
- **mtime** (*datetime.datetime*) –
- **param** (*Optional[str]*) –

Return type None

parsley.syncengine.sync module

Find the synchronization implementation in *Sync* and many other stuff in submodules.

class parsley.syncengine.sync.**Sync**(**objlist*, *name*, *interval='0s'*, *warn_after='7d'*, *warn_interval='7d'*, *benchmark_data_size=30*)

Bases: object

The synchronization implementation.

Parsley configuration files typically create some instances of this class.

The entire synchronization work takes place by all hook methods, hooked by all specified aspects. Read *Sync.executeevent()* for more details.

Parameters

- **name** (*str*) – The name for this synchronization.

- **interval** (*str*) – The synchronization interval (the engine will skip this synchronization, if the last successful run is not at least this interval ago.
- **warn_after** (*str*) – The interval after which a warning is logged when no successful run took place.
- **warn_interval** (*str*) – The interval for subsequent warnings when no successful run took place.
- **benchmark_data_size** (*int*) – The maximum capacity used for collecting benchmark data (in MB).
- **objlist** (*Union[parsley.filesystem.abstractfilesystem.Filesystem, parsley.aspect.abstractaspect.Aspect, parsley.preparation.abstractpreparation.Preparation]*) –

class EventHandler(*event*, *func*, *funcargs*)

Bases: object

A data structure that represents one hooked instance of a hook method (i.e. bound to a certain *funcargs*, including the filesystem).

Parameters

- **event** (*str*) –
- **func** (*Callable*) –
- **funcargs** (*Dict[str, Optional[Any]]*) –

_execute_helper(*runtime*, *preparations*)

Parameters

- **runtime** (parsley.runtime.runtime.RuntimeData) –
- **preparations** (*List[parsley.preparation.abstractpreparation.Preparation]*) –

Return type None

_get_eventhandlers(*event*)

Returns a list of event handlers for an event.

Parameters **event** (*str*) –

Return type List[*parsley.syncengine.sync.Sync.EventHandler*]

_hook_aspect(*aspect*, *fss*)

Request to hook program logic into certain events. Used internally. See parsley.aspect.abstractaspect.hook about how to hook custom synchronization logic.

Parameters

- **aspect** (parsley.aspect.abstractaspect.Aspect) –
- **fss** (*List[parsley.filesystem.abstractfilesystem.Filesystem]*) –

Return type None

execute(*runtime*, *force=False*)

Used by the parsley engine for executing the sync run.

Parameters

- **runtime** (parsley.runtime.runtime.RuntimeData) –

- **force** (*bool*) –

Return type bool

executeevent(*event*, *eventcontext*, ***eventparams*)

Executes an event by name. This means executing the aspect hooks and enforcing the ordering rules.

The technical realization of the workflow bases on an event-based execution engine. The events (in parsley.syncengine.common.SyncEvent) define the workflow stages of the engine. The engine essentially executes aspect hook methods (see parsley.aspect.abstractaspect.Aspect) on filesystems (see parsley.filesystem.abstractfilesystem.Filesystem) for each event.

The entire synchronization behavior is implemented by a horde of builtin aspects and optionally also custom ones. To be exact: The configuration for a synchronization task specifies the synchronization behavior by specifying a list of aspect instances. The list of aspects can be specified and configured for each filesystem individually and globally (which is equal to assigning it to each filesystem).

Whenever an aspect is assigned to one filesystem by that means, it hooks some methods into some events for that filesystem.

The order of the aspect hook executions is controlled by the symbol list *after*, *provides* and *before*. Each is a list of string symbols, e.g. *["foo", "bar", "baz"]* (you can often also write one comma-separated string like *"foo,bar,baz"*). The order of execution is chosen in a way it strictly follows this rules:

- A hook method is never executed when it contains one symbol in *after* that also exists in a *provides* list of any hook that is not yet executed (use it for after-dependencies; the typical ones).
- A hook method is never executed when it contains one symbol in *provides* that also exists in a *before* list of any hook that is not yet executed (use it for before-dependencies).

Example: A hook with *after="",provides="foo",before=""* will be executed before a one with *after="foo",provides="bar",before=""* and after a one with *after="",provides="baz",before="foo"*.

Specifying those three lists for each aspect hook method implies the execution order that must be used whenever certain hook methods depend on results or actions of other hook methods.

@note For each hook, the function name itself is always implicitly added to *provides* by the engine.

Read the manual for a high-level description of the workflow.

For advanced usage in special situations:

- A symbol in *after* and *before* can begin with a *. If so, it does not raise a validation error if this symbol is not available at all.

Parameters

- **event** (*str*) – The event to execute. See parsley.syncengine.common.SyncEvent.
- **eventcontext** (parsley.syncengine.syncruntime.SyncEventRuntime) – The parsley.syncengine.syncruntime.SyncEventRuntime control interface for this execution.
- **eventparams** (*str*) – Additional keyword parameters are added to *eventcontext* (convenience).

Return type None

sync_directory(*path*)

Executes the sub workflow for synchronizing one directory.

Parameters **path** (*str*) –

Return type None

parsley.syncengine.syncruntime module

Runtime additions for a sync event execution.

class parsley.syncengine.syncruntime.**SyncEventRuntime**(*path*)
Bases: object

Control interface for hook methods in parsley.aspect.abstractaspect.Aspect implementations.

Also holds all kinds of data for one run of parsley.syncengine.sync.Sync.sync_directory.

Dynamically extends *parsley.syncengine.syncruntime.SyncRuntime*.

Parameters **path** (*str*) –

add_conflict(*fs*, *comment=None*)
Adds a new conflict.

Parameters

- **fs** (parsley.filesystem.abstractfilesystem.Filesystem) –
- **comment** (*Optional[str]*) –

Return type None

get_conflicts()
Returns the list of all recognized conflicts.

Return type List[Tuple[*parsley.filesystem.abstractfilesystem.Filesystem*, Optional[str]]]

had_conflicts()
Returns if there are or were any conflicts.

Return type bool

is_update_marked_bad()
Returns if the update is marked as bad.

Return type bool

is_update_set_skipped()
Returns if the update process is markes to skip (by skip_update).

Return type bool

mark_update_bad()
Marks the current update as bad.

Return type None

promote_master_filesystem(*fs*, *key*)
Elects a master filesystem.

Parameters

- **fs** (parsley.filesystem.abstractfilesystem.Filesystem) – The filesystem.
- **key** (*Tuple*) – A tuple of numbers of comparison - the highest value wins.

Return type None

remove_conflict(*i*)
Removes a conflict by index.

Parameters **i** (*int*) –

Return type None

skip_master_filesystem_promotion()
Elects 'SKIP' filesystem with highest key. This will lead to skipping the current item.

Return type None

skip_update()
Marks the update process to skip.

Return type None

class parsley.syncengine.syncruntime.**SyncRuntime**
Bases: object

Control interface for sync executions.

Dynamically extends *parsley.runtime.runtime.RuntimeData*.

_read_filelist(*fs*)

Parameters **fs** (parsley.filesystem.abstractfilesystem.Filesystem) –

Return type None

_syncruntime_init()

Return type None

_verify_filelists_cookies()

Return type None

create_directory(*fs*, *path*)
Creates a directory in a filesystem. This is a high-level function that automatically handles the parsley bookkeeping and have some safeguards.

Parameters

- **fs** (parsley.filesystem.abstractfilesystem.Filesystem) –
- **path** (*str*) –

Return type None

create_link(*fs*, *path*, *linktgt*, *, *forcedupdate=False*)
Creates a link in a filesystem. This is a high-level function that automatically handles the parsley bookkeeping and have some safeguards.

Parameters

- **fs** (parsley.filesystem.abstractfilesystem.Filesystem) –
- **path** (*str*) –
- **linktgt** (*str*) –
- **forcedupdate** (*bool*) –

Return type None

delete_directory(*fs*, *path*, *, *recursive=False*)
Deletes a directory in a filesystem. This is a high-level function that automatically handles the parsley bookkeeping and have some safeguards.

Parameters

- **fs** (parsley.filesystem.abstractfilesystem.Filesystem) –

- **path** (*str*) –
- **recursive** (*bool*) –

Return type None

delete_file(*fs*, *path*)
Deletes a file in a filesystem. This is a high-level function that automatically handles the parsley bookkeeping and have some safeguards.

Parameters

- **fs** (parsley.filesystem.abstractfilesystem.Filesystem) –
- **path** (*str*) –

Return type None

delete_item(*fs*, *path*)
Deletes an item (file, directory, . . .) in a filesystem. This is a high-level function that automatically handles the parsley bookkeeping and have some safeguards.

Parameters

- **fs** (parsley.filesystem.abstractfilesystem.Filesystem) –
- **path** (*str*) –

Return type None

delete_link(*fs*, *path*)
Deletes a link in a filesystem. This is a high-level function that automatically handles the parsley bookkeeping and have some safeguards.

Parameters

- **fs** (parsley.filesystem.abstractfilesystem.Filesystem) –
- **path** (*str*) –

Return type None

getfscookie(*fs*)
Gets the cookie of a filesystem (for checking if it was completely purged/replaced between sync runs)

Parameters **fs** (parsley.filesystem.abstractfilesystem.Filesystem) –

Return type str

getinfo_current_exists(*fs*, *path*)
Gets the info from the parsley bookkeeping, if an entry with given path current exists.

Parameters

- **fs** (parsley.filesystem.abstractfilesystem.Filesystem) –
- **path** (*str*) –

Return type bool

getinfo_current_ftype(*fs*, *path*)
Gets the info from the parsley bookkeeping, which type an entry with given path has currently. The type is one of parsley.syncengine.common.EntryType.

Parameters

- **fs** (parsley.filesystem.abstractfilesystem.Filesystem) –

- **path** (*str*) –

Return type str

getinfo_current_gettags (*fs*, *path*)
Gets the tags this entry got in the last run. See also setinfo_current_addtag().

Parameters

- **fs** (parsley.filesystem.abstractfilesystem.Filesystem) –
- **path** (*str*) –

Return type List[str]

getinfo_current_mtime (*fs*, *path*)
Gets the info from the parsley bookkeeping, which modification time an entry with given path has currently.

Parameters

- **fs** (parsley.filesystem.abstractfilesystem.Filesystem) –
- **path** (*str*) –

Return type datetime.datetime

getinfo_current_param (*fs*, *path*)
Gets the info from the parsley bookkeeping, which param an entry with given path has currently. The param is used for storing the target path of a symlink, but is designed to be a general purpose field.

Parameters

- **fs** (parsley.filesystem.abstractfilesystem.Filesystem) –
- **path** (*str*) –

Return type str

getinfo_current_size (*fs*, *path*)
Gets the info from the parsley bookkeeping, which size an entry with given path has currently.

Parameters

- **fs** (parsley.filesystem.abstractfilesystem.Filesystem) –
- **path** (*str*) –

Return type int

getinfo_lastrun_exists (*fs*, *path*)
Gets the info from the parsley bookkeeping, if an entry with given path existed after last run.

Parameters

- **fs** (parsley.filesystem.abstractfilesystem.Filesystem) –
- **path** (*str*) –

Return type bool

getinfo_lastrun_ftype (*fs*, *path*)
Gets the info from the parsley bookkeeping, which type an entry with given path had after last run. The type is one of parsley.syncengine.common.EntryType.

Parameters

- **fs** (parsley.filesystem.abstractfilesystem.Filesystem) –
- **path** (*str*) –

Return type str

getinfo_lastrun_gettags (*fs*, *path*)
Gets the tags this entry currently got. See also setinfo_current_addtag().

Parameters

- **fs** (parsley.filesystem.abstractfilesystem.Filesystem) –
- **path** (*str*) –

Return type List[str]

getinfo_lastrun_mtime (*fs*, *path*)
Gets the info from the parsley bookkeeping, which modification time an entry with given path had after last run.

Parameters

- **fs** (parsley.filesystem.abstractfilesystem.Filesystem) –
- **path** (*str*) –

Return type datetime.datetime

getinfo_lastrun_param (*fs*, *path*)
Gets the info from the parsley bookkeeping, which param an entry with given path had after last run. The param is used for storing the target path of a symlink, but is designed to be a general purpose field.

Parameters

- **fs** (parsley.filesystem.abstractfilesystem.Filesystem) –
- **path** (*str*) –

Return type str

getinfo_lastrun_size (*fs*, *path*)
Gets the info from the parsley bookkeeping, which size an entry with given path had after last run.

Parameters

- **fs** (parsley.filesystem.abstractfilesystem.Filesystem) –
- **path** (*str*) –

Return type int

logcreate (*fs*, *subject*, *verb='created'*, *comment=''*, *symbol='+'*, *severity=3*)
Logs events around item creation.

Parameters

- **fs** (parsley.filesystem.abstractfilesystem.Filesystem) –
- **subject** (*str*) –
- **verb** (*str*) –
- **comment** (*str*) –
- **symbol** (*str*) –
- **severity** (*int*) –

Return type None

logproblem (*fs*, *subject*, *verb='has problems'*, *comment=''*, *symbol='E'*, *severity=6*)
Logs events around problems.

Parameters

- **fs** (parsley.filesystem.abstractfilesystem.Filesystem) –
- **subject** (*str*) –
- **verb** (*str*) –
- **comment** (*str*) –
- **symbol** (*str*) –

Return type None

logremove (*fs*, *subject*, *verb='removed'*, *comment=''*, *symbol='-'*, *severity=5*)
Logs events around item removal.

Parameters

- **fs** (parsley.filesystem.abstractfilesystem.Filesystem) –
- **subject** (*str*) –
- **verb** (*str*) –
- **comment** (*str*) –
- **symbol** (*str*) –

Return type None

logupdate (*fs*, *subject*, *verb='updated'*, *comment=''*, *symbol='*'*, *severity=4*)
Logs events around updates of exiting items.

Parameters

- **fs** (parsley.filesystem.abstractfilesystem.Filesystem) –
- **subject** (*str*) –
- **verb** (*str*) –
- **comment** (*str*) –
- **symbol** (*str*) –

Return type None

setinfo_current_addtag (*fs*, *path*, *tag*)
Adds a tag to this entry. Tags are strings that can be set and retrieved from different places in order to exchange state information. For example, the engine sets the tag *'S+'* to an entry after successful synchronization.

Parameters

- **fs** (parsley.filesystem.abstractfilesystem.Filesystem) –
- **path** (*str*) –
- **tag** (*str*) –

Return type None

setinfo_removefile (*fs*, *path*)
Mark a file as removed for a given path in the parsley bookkeeping.

Parameters

- **fs** (parsley.filesystem.abstractfilesystem.Filesystem) –

- **path** (*str*) –

Return type None

setinfo_updatefile(*fs*, *path*, *ftype*, *size=None*, *mtime=None*, *param=None*)
Updates the info stored for a given path in the parsley bookkeeping.

Parameters

- **fs** (parsley.filesystem.abstractfilesystem.Filesystem) –
- **path** (*str*) –
- **ftype** (*str*) –
- **size** (*Optional[int]*) –
- **mtime** (*Optional[datetime.datetime]*) –
- **param** (*Optional[str]*) –

Return type None

write_file(*fs*, *source*, *path*, *, *verifier=<function SyncRuntime.<lambda>>*, *forcedupdate=False*)
Creates a file in a filesystem with some content. This is a high-level function that automatically handles the parsley bookkeeping and have some safeguards.

Parameters

- **fs** (parsley.filesystem.abstractfilesystem.Filesystem) –
- **source** (*str*) –
- **path** (*str*) –
- **verifier** (*Callable*) –
- **forcedupdate** (*bool*) –

Return type Tuple[datetime.datetime, int]

Module contents

Find the synchronization implementation in *parsley.syncengine.sync.Sync* and many other stuff in submodules.

parsley.tools package

Submodules

parsley.tools.common module

Tools for very common jobs.

parsley.tools.common.**abspath**(*path*)
A variant of os.path.abspath that does what it is used for in all operating systems.

Parameters **path** (*str*) –

Return type str

parsley.tools.common.**call**(**cmdline*, *shell=False*, *decode=True*, *errorstring=None*)
Executes an external process and returns a tuple of returnvalue and program output.

Parameters

- **cmdline** (*Union[List[str], str]*) –
- **shell** (*bool*) –
- **decode** (*bool*) –
- **errorstring** (*Optional[AnyStr]*) –

Return type Union[AnyStr, Tuple[int, AnyStr]]

parsley.tools.common.**lock**(*fle*, ***, *pidless=False*, *lockpid=None*)

Locks a lockfile for synchronization. @parm pidless: If to return a lock not bound to a process. If yes, it must be unlocked and refreshed every 10

minutes.

@parm lockpid: The process id to lock with, if not the own one. This process should live over the entire timespan.

Parameters

- **fle** (*str*) –
- **pidless** (*bool*) –
- **lockpid** (*Optional[int]*) –

Return type bool

parsley.tools.common.**unlock**(*fle*)

Unlocks a lockfile.

Parameters **fle** (*str*) –

Return type None

parsley.tools.infssyncmanageconflicts module

parsley.tools.infssyncmanageconflicts.**run**()

Return type None

parsley.tools.networking module

Parsley networking features.

parsley.tools.networking.**translate_parsley_portforwarding**(*machine*, *port*, *runtime*)

Provides functionality for using parsley through an ssh remote port forwarding tunnel.

Parameters

- **machine** (*str*) –
- **port** (*int*) –
- **runtime** (parsley.syncengine.syncruntime.SyncRuntime) –

Return type Tuple[str, int]

parsley.tools.pathacceptors module

Default implementation for path acceptors as e.g. used in *parsley.aspect.applypathacceptor.ApplyPathAcceptor*.

parsley.tools.pathacceptors.**builtinpathacceptor**(*path*, *fs*)
A path acceptor that skips parsley control stuff. It is always used.

Parameters

- **path** (*str*) –
- **fs** (parsley.filesystem.abstractfilesystem.Filesystem) –

Return type bool

parsley.tools.pathacceptors.**defaultpathacceptor**(*path*, *fs*)
A path acceptor that skips file names with some known 'backup pattern', like ending with ~.

Parameters

- **path** (*str*) –
- **fs** (parsley.filesystem.abstractfilesystem.Filesystem) –

Return type bool

parsley.tools.sshchangenotifyproxy module

SSH remote proxy for monitoring and notifying file changes.

parsley.tools.sshxattrproxy module

SSH remote proxy for exchange of file metadata.

class parsley.tools.sshxattrproxy.**Mainloop**
Bases: object

This program is not directly used in the process of the parsley engine but is copied to a remote system and controlled via an ssh session as a proxy for doing some tasks on this remote system that cannot be done via an sshfs mount.

_watchdog()

processrequest(*req*)

run()

Module contents

Auxiliary stuff, (at least mostly) decoupled from Parsley infrastructure.

15.1.2 Module contents

Parsley.

Very roughly, Parlsey consists of

- A synchronization task execution engine in *parsley.syncengine* (and *parsley.runtime*)
- Lots of program pieces that bring the actual synchronization behavior in parsley.aspect
- Filesystem implementations in *parsley.filesystem*
- Some helping hands at different places like *parsley.logger*, *parsley.preparation* or *parsley.tools*
- Graphical user interfaces in *parsley.gui*

15.2 parsley_gui module

Starts Parsley main graphical user interface.

15.3 parsley_infssync_manageconflicts module

A tool for managing filesystem entry conflicts in a *parsley.syncengine.sync.Sync* sync.

15.4 parsley_infssync_manageconflicts_gui module

A gui for managing filesystem entry conflicts in a *parsley.syncengine.sync.Sync* sync.

15.5 parsley_report_gui module

Starts parsley report gui.

PYTHON MODULE INDEX

p

parsley, 112
parsley.aspect, 47
parsley.aspect.abstractaspect, 37
parsley.aspect.applypathacceptor, 39
parsley.aspect.baseinfrastructure, 39
parsley.aspect.collectinformation, 39
parsley.aspect.conflicts, 40
parsley.aspect.defaultiteration, 40
parsley.aspect.defaults, 41
parsley.aspect.electmaster, 42
parsley.aspect.highlevelcustomization, 42
parsley.aspect.logging, 43
parsley.aspect.metadata, 43
parsley.aspect.monitorfilechanges, 44
parsley.aspect.permissions, 44
parsley.aspect.readmefile, 45
parsley.aspect.remove, 45
parsley.aspect.revisiontracking, 46
parsley.aspect.update, 47
parsley.aspect.volumepathbreadcrumb, 47
parsley.aspect.volumesyncreport, 47
parsley.config, 55
parsley.config.config, 48
parsley.config.configpiece, 55
parsley.exceptions, 55
parsley.filesystem, 64
parsley.filesystem.abstractfilesystem, 56
parsley.filesystem.local, 61
parsley.filesystem.sshfs, 63
parsley.gui, 80
parsley.gui.apidef, 75
parsley.gui.app_main, 68
parsley.gui.app_main.configitem, 64
parsley.gui.app_main.runsyncdialog, 68
parsley.gui.app_manageconflicts, 69
parsley.gui.app_report, 71
parsley.gui.app_report.logpanel, 69
parsley.gui.app_report.performancepanel, 70
parsley.gui.app_report.timelinepanel, 70
parsley.gui.aspect, 72
parsley.gui.aspect.applypathacceptor, 71
parsley.gui.aspect.defaults, 71
parsley.gui.aspect.highlevelcustomization, 71
parsley.gui.aspect.logging, 72
parsley.gui.aspect.metadata, 72
parsley.gui.aspect.permissions, 72
parsley.gui.aspect.remove, 72
parsley.gui.aspect.revisiontracking, 72
parsley.gui.filesystem, 73
parsley.gui.filesystem.local, 72
parsley.gui.filesystem.sshfs, 72
parsley.gui.gtk, 74
parsley.gui.gtk.userfeedback, 73
parsley.gui.gtk.volumeapp, 73
parsley.gui.helpers, 76
parsley.gui.icons, 75
parsley.gui.report, 77
parsley.logger, 84
parsley.logger.formatter, 81
parsley.logger.formatter.abstractlogformat, 80
parsley.logger.formatter.htmllogformat, 80
parsley.logger.formatter.plaintextlogformat, 81
parsley.logger.logger, 82
parsley.logger.loggerout, 82
parsley.logger.loggerout.abstractloggerout, 81
parsley.logger.loggerout.externalprogramloggerout, 82
parsley.logger.loggerout.filestreamloggerout, 82
parsley.preparation, 87
parsley.preparation.abstractpreparation, 84
parsley.preparation.mountpreparation,

85
parsley.preparation.preparator, 86
parsley.preparation.sshfsmountpreparation, 86
parsley.runtime, 94
parsley.runtime.commonimports, 87
parsley.runtime.datastorage, 87
parsley.runtime.projectinformations, 91
parsley.runtime.returnvalue, 91
parsley.runtime.runtime, 92
parsley.runtime.successstracker, 93
parsley.syncengine, 109
parsley.syncengine.benchmark, 94
parsley.syncengine.common, 95
parsley.syncengine.core, 97
parsley.syncengine.entrylist, 98
parsley.syncengine.sync, 100
parsley.syncengine.syncruntime, 103
parsley.tools, 111
parsley.tools.common, 109
parsley.tools.infssyncmanageconflicts, 110
parsley.tools.networking, 110
parsley.tools.pathacceptors, 111
parsley.tools.sshchangenotifyproxy, 111
parsley.tools.sshxattrproxy, 111
parsley_gui, 112
parsley_infssync_manageconflicts, 112
parsley_infssync_manageconflicts_gui, 112
parsley_report_gui, 112

INDEX

Symbols

_DummyConfigItemHandler (*class in parsley.gui.app_main.configitem*), 67
_MAX_TYPICAL (*parsley.logger.logger.Severity attribute*), 83
_Metadata (*class in parsley.aspect.metadata*), 44
_MetadataStruct (*class in parsley.aspect.metadata*), 44
__about() (*parsley.gui.app_main.App method*), 68
__applydecisions() (*parsley.gui.app_manageconflicts.App method*), 69
__call_parsley_tool() (*parsley.gui.app_main.configitem.SyncTaskConfigItemHandler method*), 66
__callmanageconflicts() (*parsley.gui.app_manageconflicts.App class method*), 69
__confirm_drop_unsaved() (*parsley.gui.gtk.volumeapp.AbstractVolumeApp method*), 73
__do_add() (*parsley.gui.app_main.App method*), 68
__do_add() (*parsley.gui.app_main.configitem.FileIncludeConfigItemHandler method*), 65
__do_add_customaspect() (*parsley.gui.app_main.App method*), 68
__do_add_fileinclude() (*parsley.gui.app_main.App method*), 68
__do_add_logger() (*parsley.gui.app_main.App method*), 68
__do_add_pythonimport() (*parsley.gui.app_main.App method*), 68
__do_add_synctask() (*parsley.gui.app_main.App method*), 68
__do_config() (*parsley.gui.app_main.configitem.SyncTaskConfigItemHandler method*), 67
__do_conflicts() (*parsley.gui.app_main.configitem.SyncTaskConfigItemHandler method*), 67
__do_editcode() (*parsley.gui.app_main.configitem.CustomAspectConfigItemHandler method*), 65
__do_remove() (*parsley.gui.app_main.configitem.CustomAspectConfigItemHandler method*), 65
__do_remove() (*parsley.gui.app_main.configitem.FileIncludeConfigItemHandler method*), 65
__do_remove() (*parsley.gui.app_main.configitem.LoggerConfigItemHandler method*), 66
__do_remove() (*parsley.gui.app_main.configitem.PythonImportConfigItemHandler method*), 66
__do_remove() (*parsley.gui.app_main.configitem.SyncTaskConfigItemHandler method*), 67
__do_report() (*parsley.gui.app_main.configitem.SyncTaskConfigItemHandler method*), 67
__do_run() (*parsley.gui.app_main.configitem.SyncTaskConfigItemHand method*), 67
__draw() (*parsley.gui.app_report.timelinepanel.Timeline method*), 70
__fill_form() (*parsley.gui.app_report.performancepanel.PerformancePanel method*), 70
__getdecisions() (*parsley.gui.app_manageconflicts.App method*), 69
__gethandler_edit_aspect() (*parsley.gui.app_main.configitem.SyncTaskConfigItemHandler method*), 67
__gethandler_edit_aspect_param() (*parsley.gui.app_main.configitem.SyncTaskConfigItemHandler method*), 67
__gethandler_edit_attr() (*parsley.gui.app_main.configitem.LoggerConfigItemHandler method*), 66
__gethandler_edit_attr() (*parsley.gui.app_main.configitem.PythonImportConfigItemHandler method*), 66
__gethandler_edit_boolattr() (*pars-*

ley.gui.app_main.configitem.LoggerConfigItemHandler method), 66
__gethandler_edit_filesystem() (*parsley.gui.app_main.configitem.SyncTaskConfigItemHandler method*), 67
__gethandler_edit_filesystem_aspect() (*parsley.gui.app_main.configitem.SyncTaskConfigItemHandler method*), 67
__gethandler_edit_filesystem_aspect_param() (*parsley.gui.app_main.configitem.SyncTaskConfigItemHandler method*), 67
__gethandler_edit_filesystem_param() (*parsley.gui.app_main.configitem.SyncTaskConfigItemHandler method*), 67
__gethandler_edit_formatter() (*parsley.gui.app_main.configitem.LoggerConfigItemHandler method*), 66
__gethandler_edit_formatter_param() (*parsley.gui.app_main.configitem.LoggerConfigItemHandler method*), 66
__gethandler_edit_general() (*parsley.gui.app_main.configitem.SyncTaskConfigItemHandler method*), 67
__gethandler_edit_loggerout() (*parsley.gui.app_main.configitem.LoggerConfigItemHandler method*), 66
__gethandler_edit_loggerout_param() (*parsley.gui.app_main.configitem.LoggerConfigItemHandler method*), 66
__gethandler_edit_param() (*parsley.gui.app_main.configitem.SyncTaskConfigItemHandler method*), 67
__gethandler_edit_preparation() (*parsley.gui.app_main.configitem.SyncTaskConfigItemHandler method*), 67
__gethandler_edit_preparation_param() (*parsley.gui.app_main.configitem.SyncTaskConfigItemHandler method*), 67
__handle_dialogstate() (*parsley.gui.app_main.runsyncdialog.RunSyncDialog method*), 68
__handle_queryresult() (*parsley.gui.app_report.performancepanel.PerformancePanel method*), 70
__handle_spinner() (*parsley.gui.app_report.performancepanel.PerformancePanel method*), 70
__handleclose() (*parsley.gui.gtk.volumeapp.AbstractVolumeApp method*), 73
__help() (*parsley.gui.app_main.App method*), 68
__listconflicts() (*parsley.gui.app_manageconflicts.App class method*), 69
__loadresults() (*parsley.gui.app_report.performancepanel.PerformancePanel method*), 70
__opendir() (*parsley.gui.gtk.volumeapp.AbstractVolumeApp method*), 73
__opendirdialog() (*parsley.gui.gtk.volumeapp.AbstractVolumeApp method*), 73
__openfile() (*parsley.gui.app_main.App method*), 68
__openfiledialog() (*parsley.gui.app_main.App method*), 68
__refreshevents() (*parsley.gui.report.PerformanceData.Loader class method*), 78
__refresh() (*parsley.gui.app_report.timelinepanel.TimelinePanel method*), 71
__resolveconflict() (*parsley.gui.app_manageconflicts.App class method*), 69
_add_action_button() (*parsley.gui.gtk.volumeapp.AbstractVolumeApp method*), 73
_add_aspect() (*parsley.gui.app_main.configitem.SyncTaskConfigItemHandler method*), 67
_add_param() (*parsley.gui.app_main.configitem.ConfigItemHandler method*), 64
_add_preparation() (*parsley.gui.app_main.configitem.SyncTaskConfigItemHandler method*), 67
_ask_and_remove() (*parsley.gui.app_main.configitem.ConfigItemHandler method*), 64
_aspectlist() (*parsley.config.config.ParsleyConfiguration.Sync method*), 53
_beforeelectmaster() (*parsley.aspect.monitorfilechanges.MonitorFileChanges method*), 44
_beginsync() (*parsley.aspect.monitorfilechanges.MonitorFileChanges method*), 44
_change_type() (*parsley.gui.app_main.configitem.ConfigItemHandler method*), 64
_checkmountpointempty() (*parsley.preparation.mountpreparation.MountPreparation static method*), 85
_cleanup() (*parsley.gui.app_main.configitem.ConfigItem method*), 64
_closesync() (*parsley.aspect.monitorfilechanges.MonitorFileChanges*

method), 44
_create_control_filesystem() (*parsley.filesystem.abstractfilesystem.Filesystem method*), 56
_create_control_filesystem() (*parsley.filesystem.local.LocalFilesystem method*), 61
_createemptyconfigifneeded() (*in module parsley.syncengine.core*), 98
_edit_attr() (*parsley.gui.app_main.configitem.ConfigItemHandler method*), 64
_edit_boolattr() (*parsley.gui.app_main.configitem.ConfigItemHandler method*), 64
_edit_param() (*parsley.gui.app_main.configitem.ConfigItemHandler method*), 65
_execute_helper() (*parsley.syncengine.sync.Sync method*), 101
_get_eventhandlers() (*parsley.syncengine.sync.Sync method*), 101
_get_rootpath() (*parsley.runtime.datastorage.SyncVolumeStorageDepartment method*), 89
_get_rootpath() (*parsley.runtime.datastorage.SystemStorageDepartment method*), 90
_gettemp() (*parsley.filesystem.local.LocalFilesystem method*), 61
_has_unsaved_changes() (*parsley.gui.app_manageconflicts.App method*), 69
_has_unsaved_changes() (*parsley.gui.app_report.App method*), 71
_has_unsaved_changes() (*parsley.gui.gtk.volumeapp.AbstractVolumeApp method*), 73
_hook_aspect() (*parsley.syncengine.sync.Sync method*), 101
_logandthrowexception() (*parsley.preparation.preparator.Preparator static method*), 86
_logcreate() (*parsley.aspect.logging.Logging method*), 43
_logproblem() (*parsley.aspect.logging.Logging method*), 43
_logremove() (*parsley.aspect.logging.Logging method*), 43
_logupdate() (*parsley.aspect.logging.Logging method*), 43
_mtimes_equal() (*parsley.aspect.remove.DetectRemoval static method*), 46
_populate() (*parsley.gui.app_main.App method*), 68
_populate() (*parsley.gui.app_manageconflicts.App method*), 69
_populate() (*parsley.gui.app_report.App method*), 71
_populate() (*parsley.gui.app_report.logpanel.LogPanel method*), 69
_populate() (*parsley.gui.app_report.performancepanel.PerformancePa method*), 70
_populate() (*parsley.gui.app_report.timelinepanel.TimelinePanel method*), 71
_populate() (*parsley.gui.gtk.volumeapp.AbstractVolumeApp method*), 73
_read_filelist() (*parsley.syncengine.syncruntime.SyncRuntime method*), 104
_remove_aspect() (*parsley.gui.app_main.configitem.SyncTaskConfigItemHandler method*), 67
_remove_param() (*parsley.gui.app_main.configitem.ConfigItemHandler method*), 65
_remove_preparation() (*parsley.gui.app_main.configitem.SyncTaskConfigItemHandler method*), 67
_set_application_title() (*parsley.gui.gtk.volumeapp.AbstractVolumeApp method*), 73
_set_body() (*parsley.gui.gtk.volumeapp.AbstractVolumeApp method*), 73
_set_root_config() (*parsley.gui.app_main.configitem.ConfigItemHandler method*), 65
_set_showbody_button() (*parsley.gui.app_main.configitem.ConfigItem method*), 64
_set_welcome_text() (*parsley.gui.gtk.volumeapp.AbstractVolumeApp method*), 73
_setperms() (*parsley.aspect.permissions.ApplyPermissions static method*), 45
_syncruntime_init() (*parsley.syncengine.syncruntime.SyncRuntime method*), 104
_toplevel_window() (*parsley.gui.app_main.configitem.ConfigItemHandler property*), 65
_translate_remote_time_to_local() (*parsley.filesystem.local.LocalFilesystem method*), 61
_translator() (*in module parsley.gui.helpers*), 76
_trigger_add_item() (*parsley.gui.app_main.configitem.ConfigItem method*), 64

_trigger_reload() (*parsley.gui.app_main.configitem.ConfigItem method*), 64
_updatefile_helper() (*parsley.aspect.revisiontracking.RevisionTracking static method*), 46
_verify_filelists_cookies() (*parsley.syncengine.syncruntime.SyncRuntime method*), 104
_waitanswer() (*parsley.filesystem.sshfs.SshfsFilesystem._SshXattrProxy method*), 63
_watchdog() (*parsley.tools.sshxattrproxy.Mainloop method*), 111

A

abspath() (*in module parsley.tools.common*), 109
AbstractVolumeApp (*class in parsley.gui.gtk.volumeapp*), 73
add() (*parsley.gui.report.PerformanceData.Table method*), 79
add_conflict() (*parsley.syncengine.syncruntime.SyncEventRuntime method*), 103
add_data_available_changed_handler() (*parsley.gui.report.PerformanceData.Loader method*), 78
add_item() (*parsley.gui.app_main.configitem.ConfigItemListener method*), 65
add_listener() (*parsley.gui.app_main.configitem.ConfigItem method*), 64
add_query_result_available_changed_handler() (*parsley.gui.report.PerformanceData.Loader method*), 78
addaspect() (*parsley.config.config.ParsleyConfiguration method*), 53
addtag() (*parsley.syncengine.entrylist.CombinedEntryList method*), 98
addtag() (*parsley.syncengine.entrylist.EntryList method*), 99
App (*class in parsley.gui.app_main*), 68
App (*class in parsley.gui.app_manageconflicts*), 69
App (*class in parsley.gui.app_report*), 71
App.Conflict (*class in parsley.gui.app_manageconflicts*), 69
append_output() (*parsley.gui.app_main.runsyncdialog.RunSyncDialog method*), 68
ApplyPathAcceptor (*class in parsley.aspect.applypathacceptor*), 39
applypathacceptor_applypathacceptor() (*parsley.aspect.applypathacceptor.ApplyPathAcceptor method*), 39
ApplyPermissions (*class in parsley.aspect.permissions*), 44
applypermissions_setdirperms() (*parsley.aspect.permissions.ApplyPermissions method*), 45
applypermissions_setfileperms() (*parsley.aspect.permissions.ApplyPermissions method*), 45
Aspect (*class in parsley.aspect.abstractaspect*), 37

B

BaseInfrastructure (*class in parsley.aspect.baseinfrastructure*), 39
baseinfrastructure_update_directory() (*parsley.aspect.baseinfrastructure.BaseInfrastructure method*), 39
begincall() (*parsley.runtime.successtracker.SuccessTracker method*), 93
BeginSync (*parsley.syncengine.common.SyncEvent attribute*), 95
beginsync() (*parsley.syncengine.benchmark.BenchmarkRun method*), 94
beginsyncevent() (*parsley.syncengine.benchmark.BenchmarkRun method*), 94
beginsynceventhandler() (*parsley.syncengine.benchmark.BenchmarkRun method*), 94
begintime (*parsley.gui.app_report.timelinepanel.Timeline attribute*), 70
BenchmarkRun (*class in parsley.syncengine.benchmark*), 94
body_visible() (*parsley.gui.app_main.configitem.ConfigItem property*), 64
btncancel (*parsley.gui.app_main.runsyncdialog.RunSyncDialog attribute*), 68
bufferlog (*parsley.gui.app_main.runsyncdialog.RunSyncDialog attribute*), 68
builtinpathacceptor() (*in module parsley.tools.pathacceptors*), 111

C

call() (*in module parsley.tools.common*), 109
call_parsley_tool() (*in module parsley.gui.helpers*), 76
callonbeforeupdate() (*parsley.aspect.highlevelcustomization.HighLevelCustomization method*), 42
cancel() (*parsley.gui.helpers.ParsleyRunThread method*), 76
changeguides() (*in module parsley.gui.aspect.applypathacceptor*), 71

changeguides() (*in module parsley.gui.aspect.defaults*), 71
changeguides() (*in module parsley.gui.aspect.highlevelcustomization*), 71
changeguides() (*in module parsley.gui.aspect.logging*), 72
changeguides() (*in module parsley.gui.aspect.metadata*), 72
changeguides() (*in module parsley.gui.aspect.permissions*), 72
changeguides() (*in module parsley.gui.aspect.remove*), 72
changeguides() (*in module parsley.gui.aspect.revisiontracking*), 72
changeguides_readonly() (*in module parsley.gui.aspect.defaults*), 71
check_orphaned_dirs() (*parsley.aspect.remove.RemoveOrphanedDirs method*), 46
checkalive() (*parsley.filesystem.abstractfilesystem.Filesystem method*), 56
choicedialog() (*parsley.gui.gtk.userfeedback.UserFeedbackController method*), 73
cleanup_conflicts_dir() (*parsley.aspect.conflicts.TrackConflicts method*), 40
cleanup_init() (*parsley.aspect.remove.CleanupTrashBin method*), 45
cleanup_shadow() (*parsley.aspect.metadata.MetadataSynchronizationWithShadow method*), 43
cleanup_trashbin() (*parsley.aspect.remove.CleanupTrashBin method*), 45
CleanupTrashBin (*class in parsley.aspect.remove*), 45
clone() (*parsley.runtime.runtime.RuntimeData method*), 92
CloseSync (*parsley.syncengine.common.SyncEvent attribute*), 95
CmdLine (*class in parsley.syncengine.core*), 97
CollectInformation (*class in parsley.aspect.collectinformation*), 39
collectinformation_collectinfo() (*parsley.aspect.collectinformation.CollectInformation method*), 39
CombinedEntryList (*class in parsley.syncengine.entrylist*), 98
configfile_or_default() (*parsley.syncengine.core.CmdLine method*), 97
configfs() (*parsley.gui.filesystem.local.LocalFilesystemGuiCreateHelper method*), 72
configfs() (*parsley.gui.filesystem.sshfs.SshFilesystemGuiCreateHelper method*), 72
ConfigItem (*class in parsley.gui.app_main.configitem*), 64
configitem() (*parsley.gui.app_main.configitem.ConfigItemHandler property*), 65
ConfigItemHandler (*class in parsley.gui.app_main.configitem*), 64
ConfigItemListener (*class in parsley.gui.app_main.configitem*), 65
configitems_for_config() (*in module parsley.gui.app_main.configitem*), 67
configobject() (*parsley.gui.app_main.configitem.ConfigItem property*), 64
configobject() (*parsley.gui.app_main.configitem.ConfigItemHandler property*), 65
ConfigurationError, 55
CONTINUE_TOUCHED (*parsley.aspect.highlevelcustomization.HighLevelCustomization.Before attribute*), 42
continue_touched() (*parsley.aspect.highlevelcustomization.HighLevelCustomization.Before method*), 42
CONTINUE_UNTOUCHED (*parsley.aspect.highlevelcustomization.HighLevelCustomization.Before attribute*), 42
continue_untouched() (*parsley.aspect.highlevelcustomization.HighLevelCustomization.Before method*), 42
copyfile() (*parsley.filesystem.abstractfilesystem.Filesystem method*), 57
copyfile() (*parsley.filesystem.local.LocalFilesystem method*), 61
Core (*class in parsley.syncengine.core*), 97
Core._Myclass (*class in parsley.syncengine.core*), 97
Core.CoreRuntime (*class in parsley.syncengine.core*), 97
create_directory() (*parsley.syncengine.syncruntime.SyncRuntime method*), 104
create_link() (*parsley.syncengine.syncruntime.SyncRuntime method*), 104
createdirs() (*parsley.filesystem.abstractfilesystem.Filesystem method*), 57
createdirs() (*parsley.filesystem.local.LocalFilesystem method*), 61
(*pars-*

ley.filesystem.abstractfilesystem.Filesystem method), 57
createlink() (*parsley.filesystem.local.LocalFilesystem method*), 61
CustomAspectConfigItemHandler (*class in parsley.gui.app_main.configitem*), 65
customaspects() (*parsley.config.config.ParsleyConfiguration property*), 54

D

data_available() (*parsley.gui.report.PerformanceData.Loader property*), 78
datadir_or_default() (*parsley.syncengine.core.CmdLine method*), 97
DEBUG (*parsley.logger.logger.Severity attribute*), 83
DEBUGVERBOSE (*parsley.logger.logger.Severity attribute*), 83
DefaultBase (*class in parsley.aspect.defaults*), 41
DefaultBaseBare (*class in parsley.aspect.defaults*), 41
DefaultIteration (*class in parsley.aspect.defaultiteration*), 40
defaultiteration_listdir() (*parsley.aspect.defaultiteration.DefaultIteration method*), 40
defaultpathacceptor() (*in module parsley.tools.pathacceptors*), 111
DefaultRemove (*class in parsley.aspect.remove*), 45
defaultremove_file_link() (*parsley.aspect.remove.DefaultRemove method*), 45
DefaultRemoveDirs (*class in parsley.aspect.remove*), 45
defaultremovedirs_removedir() (*parsley.aspect.remove.DefaultRemoveDirs method*), 45
DefaultSync (*class in parsley.aspect.defaults*), 41
defaultupdateitem_detectandskipupdateconflict() (*parsley.aspect.update.DefaultUpdateItems method*), 47
defaultupdateitem_skipidentical() (*parsley.aspect.update.DefaultUpdateItems method*), 47
defaultupdateitem_skipidentical_aux() (*parsley.aspect.update.DefaultUpdateItems method*), 47
defaultupdateitem_update() (*parsley.aspect.update.DefaultUpdateItems method*), 47
DefaultUpdateItems (*class in parsley.aspect.update*), 47
delete_directory() (*parsley.syncengine.syncruntime.SyncRuntime method*), 104
delete_file() (*parsley.syncengine.syncruntime.SyncRuntime method*), 105
delete_item() (*parsley.syncengine.syncruntime.SyncRuntime method*), 105
delete_link() (*parsley.syncengine.syncruntime.SyncRuntime method*), 105
detect_type_conflict() (*parsley.aspect.conflicts.DetectTypeConflicts method*), 40
DetectRemoval (*class in parsley.aspect.remove*), 46
detectremoval_elect() (*parsley.aspect.remove.DetectRemoval method*), 46
DetectTypeConflicts (*class in parsley.aspect.conflicts*), 40
Directory (*parsley.syncengine.common.EntryType attribute*), 95
DIRTY (*parsley.runtime.returnvalue.ReturnValue attribute*), 91
disable() (*parsley.preparation.abstractpreparation.Preparation method*), 84
disable() (*parsley.preparation.mountpreparation.MountPreparation method*), 85
disable() (*parsley.preparation.sshfsmountpreparation.SshfsMountPrepa method*), 86
do_get_preferred_height() (*parsley.gui.app_report.timelinepanel.Timeline method*), 70
do_get_property() (*parsley.gui.app_main.configitem.ObjectConfigNode method*), 66
do_get_property() (*parsley.gui.app_main.configitem.ParamConfigNode method*), 66
do_get_property() (*parsley.gui.app_main.runsyncdialog.RunSyncDialog method*), 68
do_get_property() (*parsley.gui.app_report.timelinepanel.Timeline method*), 70
do_get_property() (*parsley.gui.gtk.EntryDialog method*), 74
do_set_property() (*parsley.gui.app_main.configitem.ObjectConfigNode method*), 66
do_set_property() (*parsley.gui.app_main.configitem.ParamConfigNode method*), 66

do_set_property() (*parsley.gui.app_main.runsyncdialog.RunSyncDialog method*), 68
do_set_property() (*parsley.gui.app_report.timelinepanel.Timeline method*), 70
do_set_property() (*parsley.gui.gtk.EntryDialog method*), 74

E

edtaggregation (*parsley.gui.app_report.performancepanel.PerformancePanel attribute*), 70
edtaxish (*parsley.gui.app_report.performancepanel.PerformancePanel attribute*), 70
edtaxisv (*parsley.gui.app_report.performancepanel.PerformancePanel attribute*), 70
edteventhandler (*parsley.gui.app_report.performancepanel.PerformancePanel attribute*), 70
edtfilesystem (*parsley.gui.app_report.performancepanel.PerformancePanel attribute*), 70
edtpath (*parsley.gui.app_report.performancepanel.PerformancePanel attribute*), 70
edtsyncrun (*parsley.gui.app_report.performancepanel.PerformancePanel attribute*), 70
edtsynctask (*parsley.gui.app_report.performancepanel.PerformancePanel attribute*), 70
electfilebymtime() (*parsley.aspect.electmaster.ElectMasterFileByMtime method*), 42
electlinkbyparam() (*parsley.aspect.electmaster.ElectMasterLinkByTargetHistory method*), 42
ElectMasterFileByMtime (*class in parsley.aspect.electmaster*), 42
ElectMasterLinkByTargetHistory (*class in parsley.aspect.electmaster*), 42
enable() (*parsley.preparation.abstractpreparation.Preparation method*), 84
enable() (*parsley.preparation.mountpreparation.MountPreparation method*), 85
enable() (*parsley.preparation.sshfsmountpreparation.SshfsMountPreparation method*), 86
EnablingImpossibleExecutionError, 55
end() (*parsley.runtime.successtracker.SuccessTracker method*), 93
ended() (*parsley.gui.helpers.ParsleyRunThreadListener method*), 76
EndSync (*parsley.syncengine.common.SyncEvent attribute*), 96
endsync() (*parsley.syncengine.benchmark.BenchmarkRun method*), 95
endsyncevent() (*parsley.syncengine.benchmark.BenchmarkRun method*), 95
endsynceventhandler() (*parsley.syncengine.benchmark.BenchmarkRun method*), 95
endtime (*parsley.gui.app_report.timelinepanel.Timeline attribute*), 70
ensuredisabledafter() (*parsley.preparation.abstractpreparation.Preparation method*), 84
ensuredisabledbefore() (*parsley.preparation.abstractpreparation.Preparation method*), 84
ensureenabled() (*parsley.preparation.abstractpreparation.Preparation method*), 84
entry_text (*parsley.gui.gtk.EntryDialog attribute*), 74
EntryDialog (*class in parsley.gui.gtk*), 74
EntryList (*class in parsley.syncengine.entrylist*), 98
EntryType (*class in parsley.syncengine.common*), 95
environment() (*parsley.config.config.ParsleyConfiguration property*), 54
ERROR (*parsley.logger.logger.Severity attribute*), 83
ERROR (*parsley.runtime.returnvalue.ReturnValue attribute*), 91
ERROR_EXECUTION (*parsley.runtime.returnvalue.ReturnValue attribute*), 91
ERROR_INITIALIZATION (*parsley.runtime.returnvalue.ReturnValue attribute*), 91
ERROR_PREPARATION (*parsley.runtime.returnvalue.ReturnValue attribute*), 92
ERROR_UNPREPARATION (*parsley.runtime.returnvalue.ReturnValue attribute*), 92
ErrorDisablingPreparationExecutionError, 55
ErrorEnablingPreparationExecutionError, 55
ErrorExecutingExecutionError, 56
ErrorGettingPreparationStateExecutionError, 56
ErrorInitializingExecutionError, 56
eventhandlers() (*parsley.gui.report.PerformanceData property*), 79
execute() (*parsley.syncengine.core.Core static method*), 97
execute() (*parsley.syncengine.sync.Sync method*),

101
execute_only_for_master_fs() (*in module parsley.aspect.abstractaspect*), 38
execute_only_for_master_fs_filetype() (*in module parsley.aspect.abstractaspect*), 38
execute_only_for_non_master_fs() (*in module parsley.aspect.abstractaspect*), 38
execute_only_for_slave_fs_filetype() (*in module parsley.aspect.abstractaspect*), 38
execute_only_if_not_already_maximally_elevated() (*in module parsley.aspect.abstractaspect*), 38
execute_only_if_not_update_set_skipped() (*in module parsley.aspect.abstractaspect*), 38
executeevent() (*parsley.syncengine.sync.Sync method*), 102
exists() (*parsley.filesystem.abstractfilesystem.Filesystem method*), 57
exists() (*parsley.filesystem.local.LocalFilesystem method*), 61
exists() (*parsley.syncengine.entrylist.CombinedEntryList method*), 98
exists() (*parsley.syncengine.entrylist.EntryList method*), 99
ExternalProgramLoggerout (*class in parsley.logger.loggerout.externalprogramloggerout*), 82

F

File (*parsley.syncengine.common.EntryType attribute*), 95
FileIncludeConfigItemHandler (*class in parsley.gui.app_main.configitem*), 65
filesseenwriterthread() (*parsley.aspect.monitorfilechanges.MonitorFileChanges method*), 44
FilestreamLoggerout (*class in parsley.logger.loggerout.filestreamloggerout*), 82
Filesystem (*class in parsley.filesystem.abstractfilesystem*), 56
filesystemdialog() (*parsley.gui.gtk.userfeedback.UserFeedbackController method*), 73
filesystems() (*parsley.gui.report.PerformanceData property*), 79
find_volume_rootpath() (*in module parsley.gui.helpers*), 77
flush() (*parsley.logger.logger.Logger method*), 83
flush() (*parsley.logger.loggerout.abstractloggerout.Loggerout method*), 81
flush() (*parsley.logger.loggerout.externalprogramloggerout.ExternalProgramLoggerout method*), 82
flush() (*parsley.logger.loggerout.filestreamloggerout.FilestreamLoggerout method*), 82
footer() (*parsley.logger.formatter.abstractlogformat.Logformat method*), 80
footer() (*parsley.logger.formatter.htmllogformat.HtmlLogformat method*), 80
footer() (*parsley.logger.formatter.plaintextlogformat.PlaintextLogforma method*), 81
forceexecution() (*parsley.runtime.successtracker.SuccessTracker method*), 93
foundfile() (*parsley.syncengine.entrylist.CombinedEntryList method*), 98
foundfile() (*parsley.syncengine.entrylist.EntryList method*), 99
fromxml() (*parsley.config.config.ParsleyConfiguration static method*), 54
fromxml() (*parsley.config.config.ParsleyConfiguration.Aspect static method*), 48
fromxml() (*parsley.config.config.ParsleyConfiguration.CustomAspect static method*), 48
fromxml() (*parsley.config.config.ParsleyConfiguration.Filesystem static method*), 49
fromxml() (*parsley.config.config.ParsleyConfiguration.Include static method*), 49
fromxml() (*parsley.config.config.ParsleyConfiguration.LogFormatter static method*), 50
fromxml() (*parsley.config.config.ParsleyConfiguration.Logger static method*), 51
fromxml() (*parsley.config.config.ParsleyConfiguration.Loggerout static method*), 51
fromxml() (*parsley.config.config.ParsleyConfiguration.Preparation static method*), 52
fromxml() (*parsley.config.config.ParsleyConfiguration.PythonImport static method*), 52
fromxml() (*parsley.config.config.ParsleyConfiguration.Sync static method*), 53

G

get_absolute_path() (*parsley.config.config.ParsleyConfiguration.Include method*), 50
get_conflicts() (*parsley.syncengine.syncruntime.SyncEventRuntime method*), 103
get_control_filesystem() (*parsley.filesystem.abstractfilesystem.Filesystem method*), 57
get_css_classname() (*parsley.gui.app_main.configitem._DummyConfigItemHandler method*), 67
get_css_classname() (*parsley.gui.app_main.configitem.ConfigItemHandler method*), 65
get_css_classname() (*parsley.gui.app_main.configitem.CustomAspectConfigItemHandler*

method), 65
get_css_classname() (*parsley.gui.app_main.configitem.FileIncludeConfigItemHandler method*), 65
get_css_classname() (*parsley.gui.app_main.configitem.LoggerConfigItemHandler method*), 66
get_css_classname() (*parsley.gui.app_main.configitem.PythonImportConfigItemHandler method*), 66
get_css_classname() (*parsley.gui.app_main.configitem.SyncTaskConfigItemHandler method*), 67
get_data() (*parsley.gui.report.PerformanceData.Loader method*), 78
get_filesystem() (*parsley.runtime.datastorage.StorageDepartment method*), 87
get_filesystem() (*parsley.runtime.datastorage.SyncVolumeStorageDepartment method*), 89
get_filesystem() (*parsley.runtime.datastorage.SystemStorageDepartment method*), 90
get_header_text() (*parsley.gui.app_main.configitem._DummyConfigItemHandler method*), 67
get_header_text() (*parsley.gui.app_main.configitem.ConfigItemHandler method*), 65
get_header_text() (*parsley.gui.app_main.configitem.CustomAspectConfigItemHandler method*), 65
get_header_text() (*parsley.gui.app_main.configitem.FileIncludeConfigItemHandler method*), 65
get_header_text() (*parsley.gui.app_main.configitem.LoggerConfigItemHandler method*), 66
get_header_text() (*parsley.gui.app_main.configitem.PythonImportConfigItemHandler method*), 66
get_header_text() (*parsley.gui.app_main.configitem.SyncTaskConfigItemHandler method*), 67
get_htmlcolor_for_severity() (*parsley.logger.formatter.htmllogformat.HtmlLogformat static method*), 80
get_metadata_from_file() (*parsley.aspect.metadata._Metadata static method*), 44
get_metadata_from_shadow() (*parsley.aspect.metadata._Metadata static method*), 44
get_parent_filesystem() (*parsley.filesystem.abstractfilesystem.Filesystem method*), 57
get_query_result() (*parsley.gui.report.PerformanceData.Loader method*), 78
get_report() (*parsley.syncengine.benchmark.BenchmarkRun method*), 95
get_retval() (*parsley.runtime.runtime.RuntimeData method*), 92
get_storage_department() (*in module parsley.runtime.datastorage*), 91
get_value_bytes() (*parsley.runtime.datastorage.StorageDepartment method*), 87
get_value_bytes() (*parsley.runtime.datastorage.SyncVolumeStorageDepartment method*), 90
get_value_bytes() (*parsley.runtime.datastorage.SystemStorageDepartment method*), 90
get_value_path() (*parsley.runtime.datastorage.SyncVolumeStorageDepartment method*), 90
get_value_path() (*parsley.runtime.datastorage.SystemStorageDepartment method*), 90
get_value_string() (*parsley.runtime.datastorage.StorageDepartment method*), 88
get_volume_paths_for_sync() (*in module parsley.gui.helpers*), 77
getaspects() (*parsley.config.config.ParsleyConfiguration.Sync method*), 53
getbool() (*in module parsley.config.configpiece*), 55
getcallable() (*in module parsley.config.configpiece*), 55
getdata() (*parsley.aspect.metadata._Metadata method*), 44
getfscookie() (*parsley.syncengine.syncruntime.SyncRuntime method*), 105
getfsname() (*in module parsley.gui.apidef*), 75
getftype() (*parsley.filesystem.abstractfilesystem.Filesystem method*), 57
getftype() (*parsley.filesystem.local.LocalFilesystem method*), 61
getftype() (*parsley.syncengine.entrylist.CombinedEntryList method*), 98
getftype() (*parsley.syncengine.entrylist.EntryList method*), 99
getfulllocalpath() (*pars-*

ley.filesystem.abstractfilesystem.Filesystem method), 57
getfulllocalpath() (*parsley.filesystem.local.LocalFilesystem method*), 61
getfulllocalpathorfallback() (*parsley.filesystem.abstractfilesystem.Filesystem method*), 58
getinfo_current_exists() (*parsley.syncengine.syncruntime.SyncRuntime method*), 105
getinfo_current_ftype() (*parsley.syncengine.syncruntime.SyncRuntime method*), 105
getinfo_current_gettags() (*parsley.syncengine.syncruntime.SyncRuntime method*), 106
getinfo_current_mtime() (*parsley.syncengine.syncruntime.SyncRuntime method*), 106
getinfo_current_param() (*parsley.syncengine.syncruntime.SyncRuntime method*), 106
getinfo_current_size() (*parsley.syncengine.syncruntime.SyncRuntime method*), 106
getinfo_lastrun_exists() (*parsley.syncengine.syncruntime.SyncRuntime method*), 106
getinfo_lastrun_ftype() (*parsley.syncengine.syncruntime.SyncRuntime method*), 106
getinfo_lastrun_gettags() (*parsley.syncengine.syncruntime.SyncRuntime method*), 107
getinfo_lastrun_mtime() (*parsley.syncengine.syncruntime.SyncRuntime method*), 107
getinfo_lastrun_param() (*parsley.syncengine.syncruntime.SyncRuntime method*), 107
getinfo_lastrun_size() (*parsley.syncengine.syncruntime.SyncRuntime method*), 107
getinputstream() (*parsley.filesystem.abstractfilesystem.Filesystem method*), 58
getinputstream() (*parsley.filesystem.local.LocalFilesystem method*), 61
getkeys() (*parsley.aspect.metadata._Metadata method*), 44
getlastsuccessfulcall() (*parsley.runtime.successtracker.SuccessTracker method*), 93
getlastsyncedmtime() (*parsley.syncengine.entrylist.CombinedEntryList method*), 98
getlastsyncedmtime() (*parsley.syncengine.entrylist.EntryList method*), 99
getlinktarget() (*parsley.filesystem.abstractfilesystem.Filesystem method*), 58
getlinktarget() (*parsley.filesystem.local.LocalFilesystem method*), 61
getlist() (*in module parsley.config.configpiece*), 55
getmtime() (*parsley.filesystem.abstractfilesystem.Filesystem method*), 58
getmtime() (*parsley.filesystem.local.LocalFilesystem method*), 61
getmtime() (*parsley.syncengine.entrylist.CombinedEntryList method*), 98
getmtime() (*parsley.syncengine.entrylist.EntryList method*), 99
getnumstring() (*in module parsley.gui.apidef*), 75
getoutputstream() (*parsley.filesystem.abstractfilesystem.Filesystem method*), 58
getoutputstream() (*parsley.filesystem.local.LocalFilesystem method*), 61
getparam() (*parsley.syncengine.entrylist.CombinedEntryList method*), 98
getparam() (*parsley.syncengine.entrylist.EntryList method*), 99
getregisteredchangeguides() (*in module parsley.gui.apidef*), 75
getregisteredfilesystemhelpers() (*in module parsley.gui.apidef*), 75
getsize() (*parsley.filesystem.abstractfilesystem.Filesystem method*), 58
getsize() (*parsley.filesystem.local.LocalFilesystem method*), 62
getsize() (*parsley.syncengine.entrylist.CombinedEntryList method*), 98
getsize() (*parsley.syncengine.entrylist.EntryList method*), 99
getsshxattrproxy() (*parsley.filesystem.sshfs.SshfsFilesystem method*), 63
getstate() (*parsley.preparation.abstractpreparation.Preparation method*), 84
getstate() (*parsley.preparation.mountpreparation.MountPreparation method*), 85
gettags() (*parsley.syncengine.entrylist.CombinedEntryList method*), 98

gettags() (*parsley.syncengine.entrylist.EntryList method*), 99
gettimedelta() (*in module parsley.config.configpiece*), 55
getxattrvalue() (*parsley.filesystem.abstractfilesystem.Filesystem method*), 58
getxattrvalue() (*parsley.filesystem.local.LocalFilesystem method*), 62
getxattrvalue() (*parsley.filesystem.sshfs.SshfsFilesystem method*), 63

H

had_conflicts() (*parsley.syncengine.syncruntime.SyncEventRuntime method*), 103
has_value() (*parsley.runtime.datastorage.StorageDepartment method*), 88
has_value() (*parsley.runtime.datastorage.SystemStorageDepartment method*), 91
hasaspect() (*parsley.config.config.ParsleyConfiguration.Sync method*), 53
header() (*parsley.logger.formatter.abstractlogformat.Logformat method*), 80
header() (*parsley.logger.formatter.htmllogformat.HtmlLogformat method*), 80
header() (*parsley.logger.formatter.plaintextlogformat.PlaintextLogformat method*), 81
HighLevelCustomization (*class in parsley.aspect.highlevelcustomization*), 42
HighLevelCustomization.BeforeUpdateEvent (*class in parsley.aspect.highlevelcustomization*), 42
HighLevelCustomization.BeforeUpdateHandlerTerminatedUnexpectedly, 42
hook() (*in module parsley.aspect.abstractaspect*), 38
HtmlLogformat (*class in parsley.logger.formatter.htmllogformat*), 80

I

IMPORTANT (*parsley.logger.logger.Severity attribute*), 83
includes() (*parsley.config.config.ParsleyConfiguration property*), 54
indicatorsteptime (*parsley.gui.app_report.timelinepanel.Timeline attribute*), 70
INFO (*parsley.logger.logger.Severity attribute*), 83
init() (*parsley.aspect.highlevelcustomization.HighLevelCustomization method*), 42
init_template() (*parsley.gui.app_main.configitem.ConfigItem method*), 64
init_template() (*parsley.gui.app_main.runsyncdialog.RunSyncDialog method*), 68
init_template() (*parsley.gui.app_report.logpanel.LogPanel method*), 69
init_template() (*parsley.gui.app_report.performancepanel.PerformancePanel method*), 70
init_template() (*parsley.gui.app_report.timelinepanel.TimelinePanel method*), 71
init_ui() (*parsley.gui.app_main.configitem.ConfigItemHandler method*), 65
init_ui() (*parsley.gui.app_main.configitem.CustomAspectConfigItemH method*), 65
init_ui() (*parsley.gui.app_main.configitem.FileIncludeConfigItemHand method*), 65
init_ui() (*parsley.gui.app_main.configitem.LoggerConfigItemHandler method*), 66
init_ui() (*parsley.gui.app_main.configitem.PythonImportConfigItemHa method*), 66
init_ui() (*parsley.gui.app_main.configitem.SyncTaskConfigItemHandle method*), 67
initialize() (*parsley.filesystem.abstractfilesystem.Filesystem method*), 58
initialize() (*parsley.filesystem.local.LocalFilesystem method*), 62
initialize() (*parsley.filesystem.sshfs.SshfsFilesystem method*), 63
initialize_late() (*parsley.filesystem.abstractfilesystem.Filesystem method*), 59
initialize_late() (*parsley.filesystem.local.LocalFilesystem method*), 62
inputdialog() (*parsley.gui.gtk.userfeedback.UserFeedbackController method*), 74
instantiate() (*parsley.config.config.ParsleyConfiguration.Aspect method*), 48
instantiate() (*parsley.config.config.ParsleyConfiguration.CustomAspect method*), 49
instantiate() (*parsley.config.config.ParsleyConfiguration.Filesystem method*), 49
instantiate() (*parsley.config.config.ParsleyConfiguration.LogFormatter*

method), 50
instantiate() (*parsley.config.config.ParsleyConfiguration.Logger method*), 51
instantiate() (*parsley.config.config.ParsleyConfiguration.Loggerout method*), 51
instantiate() (*parsley.config.config.ParsleyConfiguration.Preparation method*), 52
instantiate() (*parsley.config.config.ParsleyConfiguration.PythonImport method*), 52
instantiate() (*parsley.config.config.ParsleyConfiguration.Sync method*), 53
InvalidCommandLineError, 56
InvalidUserFeedbackController (*class in parsley.gui.apidef*), 75
is_control_filesystem() (*parsley.filesystem.abstractfilesystem.Filesystem method*), 59
is_mtime_precision_fine() (*parsley.filesystem.abstractfilesystem.Filesystem method*), 59
is_mtime_precision_fine() (*parsley.filesystem.local.LocalFilesystem method*), 62
is_update_marked_bad() (*parsley.syncengine.syncruntime.SyncEventRuntime method*), 103
is_update_set_skipped() (*parsley.syncengine.syncruntime.SyncEventRuntime method*), 103

K

key_label (*parsley.gui.app_main.configitem.ParamConfigNode attribute*), 66

L

label (*parsley.gui.app_main.configitem.ObjectConfigNode attribute*), 66
lblhead (*parsley.gui.app_main.configitem.ConfigItem attribute*), 64
Link (*parsley.syncengine.common.EntryType attribute*), 95
list_path() (*parsley.runtime.datastorage.StorageDepartment method*), 88
listdir() (*parsley.filesystem.abstractfilesystem.Filesystem method*), 59
listdir() (*parsley.filesystem.local.LocalFilesystem method*), 62
listxattrkeys() (*parsley.filesystem.abstractfilesystem.Filesystem method*), 59
listxattrkeys() (*parsley.filesystem.local.LocalFilesystem method*), 62
listxattrkeys() (*parsley.filesystem.sshfs.SshfsFilesystem method*), 63
load_common_css() (*in module parsley.gui.gtk*), 74
load_css() (*in module parsley.gui.gtk*), 75
loadcustomizations() (*parsley.aspect.highlevelcustomization.HighLevelCustomization method*), 43
LocalFilesystem (*class in parsley.filesystem.local*), 61
LocalFilesystemGuiCreateHelper (*class in parsley.gui.filesystem.local*), 72
lock() (*in module parsley.tools.common*), 110
log() (*parsley.logger.formatter.abstractlogformat.Logformat method*), 80
log() (*parsley.logger.formatter.htmllogformat.HtmlLogformat method*), 80
log() (*parsley.logger.formatter.plaintextlogformat.PlaintextLogformat method*), 81
log() (*parsley.logger.logger.Logger method*), 83
log() (*parsley.logger.loggerout.abstractloggerout.Loggerout method*), 81
log() (*parsley.logger.loggerout.externalprogramloggerout.ExternalProgra method*), 82
log() (*parsley.logger.loggerout.filestreamloggerout.FilestreamLoggerout method*), 82
log() (*parsley.runtime.runtime.RuntimeData method*), 92
LogCreate (*parsley.syncengine.common.SyncEvent attribute*), 96
logcreate() (*parsley.syncengine.syncruntime.SyncRuntime method*), 107
Logformat (*class in parsley.logger.formatter.abstractlogformat*), 80
Logger (*class in parsley.logger.logger*), 82
LoggerConfigItemHandler (*class in parsley.gui.app_main.configitem*), 65
Loggerout (*class in parsley.logger.loggerout.abstractloggerout*), 81
loggers() (*parsley.config.config.ParsleyConfiguration property*), 54
Logging (*class in parsley.aspect.logging*), 43
LogMessage (*class in parsley.logger.logger*), 82
LogMessageWithTime (*class in parsley.gui.report*), 77
LogPanel (*class in parsley.gui.app_report.logpanel*), 69
LogProblem (*parsley.syncengine.common.SyncEvent attribute*), 96
logproblem() (*pars-*

ley.syncengine.syncruntime.SyncRuntime method), 107
LogRemove (*parsley.syncengine.common.SyncEvent attribute*), 96
logremove() (*parsley.syncengine.syncruntime.SyncRuntime method*), 108
LogUpdate (*parsley.syncengine.common.SyncEvent attribute*), 96
logupdate() (*parsley.syncengine.syncruntime.SyncRuntime method*), 108

M

main() (*in module parsley.syncengine.core*), 98
Mainloop (*class in parsley.tools.sshxattrproxy*), 111
mark_dirty() (*parsley.runtime.runtime.RuntimeData method*), 93
mark_update_bad() (*parsley.syncengine.syncruntime.SyncEventRuntime method*), 103
messagedialog() (*parsley.gui.gtk.userfeedback.UserFeedbackController method*), 74
metadata_checkagainstshadow() (*parsley.aspect.metadata.MetadataSynchronization method*), 43
metadata_differs() (*parsley.aspect.metadata._Metadata static method*), 44
metadata_findhighestshadow() (*parsley.aspect.metadata.MetadataSynchronizationWithShadow method*), 43
metadata_init() (*parsley.aspect.metadata.MetadataSynchronizationWithShadow method*), 43
metadata_propagatetofilesystem_file() (*parsley.aspect.metadata.MetadataSynchronization method*), 43
metadata_propagatetoshadow_file() (*parsley.aspect.metadata.MetadataSynchronizationWithShadow method*), 43
MetadataSynchronization (*class in parsley.aspect.metadata*), 43
MetadataSynchronizationWithShadow (*class in parsley.aspect.metadata*), 43
module
    parsley, 112
    parsley.aspect, 47
    parsley.aspect.abstractaspect, 37
    parsley.aspect.applypathacceptor, 39
    parsley.aspect.baseinfrastructure, 39
    parsley.aspect.collectinformation, 39
    parsley.aspect.conflicts, 40
    parsley.aspect.defaultiteration, 40
    parsley.aspect.defaults, 41
    parsley.aspect.electmaster, 42
    parsley.aspect.highlevelcustomization, 42
    parsley.aspect.logging, 43
    parsley.aspect.metadata, 43
    parsley.aspect.monitorfilechanges, 44
    parsley.aspect.permissions, 44
    parsley.aspect.readmefile, 45
    parsley.aspect.remove, 45
    parsley.aspect.revisiontracking, 46
    parsley.aspect.update, 47
    parsley.aspect.volumepathbreadcrumb, 47
    parsley.aspect.volumesyncreport, 47
    parsley.config, 55
    parsley.config.config, 48
    parsley.config.configpiece, 55
    parsley.exceptions, 55
    parsley.filesystem, 64
    parsley.filesystem.abstractfilesystem, 56
    parsley.filesystem.local, 61
    parsley.filesystem.sshfs, 63
    parsley.gui, 80
    parsley.gui.apidef, 75
    parsley.gui.app_main, 68
    parsley.gui.app_main.configitem, 64
    parsley.gui.app_main.runsyncdialog, 68
    parsley.gui.app_manageconflicts, 69
    parsley.gui.app_report, 71
    parsley.gui.app_report.logpanel, 69
    parsley.gui.app_report.performancepanel, 70
    parsley.gui.app_report.timelinepanel, 70
    parsley.gui.aspect, 72
    parsley.gui.aspect.applypathacceptor, 71
    parsley.gui.aspect.defaults, 71
    parsley.gui.aspect.highlevelcustomization, 71
    parsley.gui.aspect.logging, 72
    parsley.gui.aspect.metadata, 72
    parsley.gui.aspect.permissions, 72
    parsley.gui.aspect.remove, 72
    parsley.gui.aspect.revisiontracking, 72
    parsley.gui.filesystem, 73
    parsley.gui.filesystem.local, 72
    parsley.gui.filesystem.sshfs, 72

parsley.gui.gtk, 74
parsley.gui.gtk.userfeedback, 73
parsley.gui.gtk.volumeapp, 73
parsley.gui.helpers, 76
parsley.gui.icons, 75
parsley.gui.report, 77
parsley.logger, 84
parsley.logger.formatter, 81
parsley.logger.formatter.abstractlogformatter, 80
parsley.logger.formatter.htmllogformat, 80
parsley.logger.formatter.plaintextlogformat, 81
parsley.logger.logger, 82
parsley.logger.loggerout, 82
parsley.logger.loggerout.abstractloggerout, 81
parsley.logger.loggerout.externalprogramloggerout, 82
parsley.logger.loggerout.filestreamloggerout, 82
parsley.preparation, 87
parsley.preparation.abstractpreparation, 84
parsley.preparation.mountpreparation, 85
parsley.preparation.preparator, 86
parsley.preparation.sshfsmountpreparation, 86
parsley.runtime, 94
parsley.runtime.commonimports, 87
parsley.runtime.datastorage, 87
parsley.runtime.projectinformations, 91
parsley.runtime.returnvalue, 91
parsley.runtime.runtime, 92
parsley.runtime.successtracker, 93
parsley.syncengine, 109
parsley.syncengine.benchmark, 94
parsley.syncengine.common, 95
parsley.syncengine.core, 97
parsley.syncengine.entrylist, 98
parsley.syncengine.sync, 100
parsley.syncengine.syncruntime, 103
parsley.tools, 111
parsley.tools.common, 109
parsley.tools.infssyncmanageconflicts, 110
parsley.tools.networking, 110
parsley.tools.pathacceptors, 111
parsley.tools.sshchangenotifyproxy, 111
parsley.tools.sshxattrproxy, 111
parsley_gui, 112
parsley_infssync_manageconflicts, 112
parsley_infssync_manageconflicts_gui, 112
parsley_report_gui, 112
MonitorFileChanges (*class in parsley.aspect.monitorfilechanges*), 44
monitorproxy() (*parsley.aspect.monitorfilechanges.MonitorFileChanges static method*), 44
MOREIMPORTANT (*parsley.logger.logger.Severity attribute*), 83
MountpointNotEmptyError, 85
MountPreparation (*class in parsley.preparation.mountpreparation*), 85
MountPreparationError, 85
move() (*parsley.filesystem.abstractfilesystem.Filesystem method*), 59
move() (*parsley.filesystem.local.LocalFilesystem method*), 62
multilineinputdialog() (*parsley.gui.gtk.userfeedback.UserFeedbackController method*), 74

N

name() (*parsley.config.config.ParsleyConfiguration.Filesystem property*), 49
name() (*parsley.config.config.ParsleyConfiguration.Sync property*), 53
notfoundfile() (*parsley.syncengine.entrylist.CombinedEntryList method*), 98
notfoundfile() (*parsley.syncengine.entrylist.EntryList method*), 100

O

ObjectConfigNode (*class in parsley.gui.app_main.configitem*), 66
openconfig() (*in module parsley.gui.helpers*), 77
output() (*parsley.gui.helpers.ParsleyRunThreadListener method*), 76

P

ParamConfigNode (*class in parsley.gui.app_main.configitem*), 66
parseparams() (*parsley.config.config.ParsleyConfiguration method*), 54
parsestring() (*parsley.config.config.ParsleyConfiguration method*), 54
parsley

module, 112
parsley.aspect
module, 47
parsley.aspect.abstractaspect
module, 37
parsley.aspect.applypathacceptor
module, 39
parsley.aspect.baseinfrastructure
module, 39
parsley.aspect.collectinformation
module, 39
parsley.aspect.conflicts
module, 40
parsley.aspect.defaultiteration
module, 40
parsley.aspect.defaults
module, 41
parsley.aspect.electmaster
module, 42
parsley.aspect.highlevelcustomization
module, 42
parsley.aspect.logging
module, 43
parsley.aspect.metadata
module, 43
parsley.aspect.monitorfilechanges
module, 44
parsley.aspect.permissions
module, 44
parsley.aspect.readmefile
module, 45
parsley.aspect.remove
module, 45
parsley.aspect.revisiontracking
module, 46
parsley.aspect.update
module, 47
parsley.aspect.volumepathbreadcrumb
module, 47
parsley.aspect.volumesyncreport
module, 47
parsley.config
module, 55
parsley.config.config
module, 48
parsley.config.configpiece
module, 55
parsley.exceptions
module, 55
parsley.filesystem
module, 64
parsley.filesystem.abstractfilesystem
module, 56
parsley.filesystem.local
module, 61
parsley.filesystem.sshfs
module, 63
parsley.gui
module, 80
parsley.gui.apidef
module, 75
parsley.gui.app_main
module, 68
parsley.gui.app_main.configitem
module, 64
parsley.gui.app_main.runsyncdialog
module, 68
parsley.gui.app_manageconflicts
module, 69
parsley.gui.app_report
module, 71
parsley.gui.app_report.logpanel
module, 69
parsley.gui.app_report.performancepanel
module, 70
parsley.gui.app_report.timelinepanel
module, 70
parsley.gui.aspect
module, 72
parsley.gui.aspect.applypathacceptor
module, 71
parsley.gui.aspect.defaults
module, 71
parsley.gui.aspect.highlevelcustomization
module, 71
parsley.gui.aspect.logging
module, 72
parsley.gui.aspect.metadata
module, 72
parsley.gui.aspect.permissions
module, 72
parsley.gui.aspect.remove
module, 72
parsley.gui.aspect.revisiontracking
module, 72
parsley.gui.filesystem
module, 73
parsley.gui.filesystem.local
module, 72
parsley.gui.filesystem.sshfs
module, 72
parsley.gui.gtk
module, 74
parsley.gui.gtk.userfeedback
module, 73
parsley.gui.gtk.volumeapp
module, 73
parsley.gui.helpers

module, 76
parsley.gui.icons
module, 75
parsley.gui.report
module, 77
parsley.logger
module, 84
parsley.logger.formatter
module, 81
parsley.logger.formatter.abstractlogformat
module, 80
parsley.logger.formatter.htmllogformat
module, 80
parsley.logger.formatter.plaintextlogformat
module, 81
parsley.logger.logger
module, 82
parsley.logger.loggerout
module, 82
parsley.logger.loggerout.abstractloggerout
module, 81
parsley.logger.loggerout.externalprogramloggerout
module, 82
parsley.logger.loggerout.filestreamloggerout
module, 82
parsley.preparation
module, 87
parsley.preparation.abstractpreparation
module, 84
parsley.preparation.mountpreparation
module, 85
parsley.preparation.preparator
module, 86
parsley.preparation.sshfsmountpreparation
module, 86
parsley.runtime
module, 94
parsley.runtime.commonimports
module, 87
parsley.runtime.datastorage
module, 87
parsley.runtime.projectinformations
module, 91
parsley.runtime.returnvalue
module, 91
parsley.runtime.runtime
module, 92
parsley.runtime.successtracker
module, 93
parsley.syncengine
module, 109
parsley.syncengine.benchmark
module, 94
parsley.syncengine.common
module, 95
parsley.syncengine.core
module, 97
parsley.syncengine.entrylist
module, 98
parsley.syncengine.sync
module, 100
parsley.syncengine.syncruntime
module, 103
parsley.tools
module, 111
parsley.tools.common
module, 109
parsley.tools.infssyncmanageconflicts
module, 110
parsley.tools.networking
module, 110
parsley.tools.pathacceptors
module, 111
parsley.tools.sshchangenotifyproxy
module, 111
parsley.tools.sshxattrproxy
module, 111
parsley_cmdline() (*in module parsley.gui.helpers*), 77
parsley_gui
module, 112
parsley_infssync_manageconflicts
module, 112
parsley_infssync_manageconflicts_gui
module, 112
parsley_report_gui
module, 112
ParsleyConfiguration (*class in parsley.config.config*), 48
ParsleyConfiguration.Aspect (*class in parsley.config.config*), 48
ParsleyConfiguration.CustomAspect (*class in parsley.config.config*), 48
ParsleyConfiguration.Filesystem (*class in parsley.config.config*), 49
ParsleyConfiguration.Include (*class in parsley.config.config*), 49
ParsleyConfiguration.LogFormatter (*class in parsley.config.config*), 50
ParsleyConfiguration.Logger (*class in parsley.config.config*), 50
ParsleyConfiguration.Loggerout (*class in parsley.config.config*), 51
ParsleyConfiguration.Preparation (*class in parsley.config.config*), 51
ParsleyConfiguration.PythonImport (*class in parsley.config.config*), 52

ParsleyConfiguration.Sync (*class in parsley.config.config*), 52
ParsleyEngineExecutionError, 56
ParsleyError, 56
ParsleyGtkBuilder (*class in parsley.gui.gtk*), 74
ParsleyRunThread (*class in parsley.gui.helpers*), 76
ParsleyRunThreadListener (*class in parsley.gui.helpers*), 76
PER_SYNC (*parsley.runtime.datastorage.StorageScope attribute*), 89
PerformanceData (*class in parsley.gui.report*), 77
PerformanceData.EventHandler (*class in parsley.gui.report*), 78
PerformanceData.Filesystem (*class in parsley.gui.report*), 78
PerformanceData.Loader (*class in parsley.gui.report*), 78
PerformanceData.Path (*class in parsley.gui.report*), 78
PerformanceData.QueryResult (*class in parsley.gui.report*), 78
PerformanceData.Sync (*class in parsley.gui.report*), 79
PerformanceData.SyncRun (*class in parsley.gui.report*), 79
PerformanceData.Table (*class in parsley.gui.report*), 79
PerformancePanel (*class in parsley.gui.app_report.performancepanel*), 70
PlaintextLogformat (*class in parsley.logger.formatter.plaintextlogformat*), 81
pnlbody (*parsley.gui.app_main.configitem.ConfigItem attribute*), 64
pnlbodyouter (*parsley.gui.app_main.configitem.ConfigItem attribute*), 64
pnlconfigbody (*parsley.gui.app_main.configitem.ConfigItem attribute*), 64
pnllog (*parsley.gui.app_report.logpanel.LogPanel attribute*), 69
pnlsyncfailed (*parsley.gui.app_main.runsyncdialog.RunSyncDialog attribute*), 68
pnlsyncing (*parsley.gui.app_main.runsyncdialog.RunSyncDialog attribute*), 68
pnlsyncsucceeded (*parsley.gui.app_main.runsyncdialog.RunSyncDialog attribute*), 68
pnltimelines (*parsley.gui.app_report.timelinepanel.TimelinePanel attribute*), 71
popover_menu() (*in module parsley.gui.gtk*), 75
ppsyncsink_electifnowhereelse() (*parsley.aspect.defaults.PullAndPurgeSyncSink method*), 41
ppsyncsink_renameexistingtonew() (*parsley.aspect.defaults.PullAndPurgeSyncSink method*), 41
ppsyncsource_removefromsource() (*parsley.aspect.defaults.PullAndPurgeSyncSource method*), 42
Preparation (*class in parsley.preparation.abstractpreparation*), 84
PreparationDisabledAfterEnablingExecutionError, 56
PreparationEnabledAfterDisablingExecutionError, 56
PreparationEnabledBeforeEnablingExecutionError, 56
Preparator (*class in parsley.preparation.preparator*), 86
prepare_trackconflicts() (*parsley.aspect.conflicts.TrackConflicts method*), 40
processrequest() (*parsley.tools.sshxattrproxy.Mainloop method*), 111
promote_master_filesystem() (*parsley.syncengine.syncruntime.SyncEventRuntime method*), 103
PullAndPurgeSyncSink (*class in parsley.aspect.defaults*), 41
PullAndPurgeSyncSource (*class in parsley.aspect.defaults*), 42
putdata() (*parsley.aspect.metadata._Metadata method*), 44
PythonImportConfigItemHandler (*class in parsley.gui.app_main.configitem*), 66
pythonimports() (*parsley.config.config.ParsleyConfiguration property*), 54

Q

query() (*parsley.gui.report.PerformanceData.Loader method*), 78
query_result_available() (*parsley.gui.report.PerformanceData.Loader property*), 78

R

read() (*parsley.syncengine.entrylist.EntryList method*), 100
readfromfile() (*parsley.filesystem.abstractfilesystem.Filesystem method*), 59
readfromfile() (*parsley.filesystem.local.LocalFilesystem method*),

62
ReadInvalidConfigurationError, 56
ReadmeFile (*class in parsley.aspect.readmefile*), 45
readmefile_writereadme() (*parsley.aspect.readmefile.ReadmeFile method*), 45
refresh() (*parsley.gui.app_main.configitem.ConfigItem method*), 64
registerchangeguide() (*in module parsley.gui.apidef*), 75
registerfilesystemhelper() (*in module parsley.gui.apidef*), 75
reload() (*parsley.gui.app_main.configitem.ConfigItemHandler method*), 65
reload() (*parsley.gui.app_main.configitem.ConfigItemListener method*), 65
remove_conflict() (*parsley.syncengine.syncruntime.SyncEventRuntime method*), 103
remove_orphaned_dirs() (*parsley.aspect.remove.RemoveOrphanedDirs method*), 46
remove_shadow() (*parsley.aspect.metadata._Metadata static method*), 44
remove_value() (*parsley.runtime.datastorage.StorageDepartment method*), 88
removeaspect() (*parsley.config.config.ParsleyConfiguration.Sync method*), 53
removedir() (*parsley.filesystem.abstractfilesystem.Filesystem method*), 59
removedir() (*parsley.filesystem.local.LocalFilesystem method*), 62
removefile() (*parsley.filesystem.abstractfilesystem.Filesystem method*), 60
removefile() (*parsley.filesystem.local.LocalFilesystem method*), 62
removefile() (*parsley.syncengine.entrylist.CombinedEntryList method*), 98
removefile() (*parsley.syncengine.entrylist.EntryList method*), 100
removelink() (*parsley.filesystem.abstractfilesystem.Filesystem method*), 60
removelink() (*parsley.filesystem.local.LocalFilesystem method*), 62
RemoveOrphanedDirs (*class in parsley.aspect.remove*), 46
request() (*parsley.filesystem.sshfs.SshfsFilesystem._SshXattrProxy method*), 63
resizedetector (*parsley.gui.app_report.timelinepanel.TimelinePanel attribute*), 71
resolveconflicts_byhint() (*parsley.aspect.conflicts.ResolveConflictsByHint method*), 40
resolveconflicts_cleanupbeforenewdir() (*parsley.aspect.update.DefaultUpdateItems method*), 47
ResolveConflictsByHint (*class in parsley.aspect.conflicts*), 40
ReturnValue (*class in parsley.runtime.returnvalue*), 91
RevisionTracking (*class in parsley.aspect.revisiontracking*), 46
revisiontracking_init() (*parsley.aspect.revisiontracking.RevisionTracking method*), 46
revisiontracking_store1() (*parsley.aspect.revisiontracking.RevisionTracking method*), 46
revisiontracking_store2() (*parsley.aspect.revisiontracking.RevisionTracking method*), 46
run() (*in module parsley.gui.app_main*), 68
run() (*in module parsley.gui.app_manageconflicts*), 69
run() (*in module parsley.gui.app_report*), 71
run() (*in module parsley.tools.infssyncmanageconflicts*), 110
run() (*parsley.gui.helpers.ParsleyRunThread method*), 76
run() (*parsley.tools.sshxattrproxy.Mainloop method*), 111
run_and_get_entry_text() (*parsley.gui.gtk.EntryDialog method*), 74
RunSyncDialog (*class in parsley.gui.app_main.runsyncdialog*), 68
RuntimeData (*class in parsley.runtime.runtime*), 92

S

save() (*parsley.syncengine.entrylist.EntryList method*), 100
saveconfig() (*in module parsley.gui.helpers*), 77
scrollview (*parsley.gui.app_report.performancepanel.PerformancePan attribute*), 70
set_metadata_to_file() (*parsley.aspect.metadata._Metadata static method*), 44
set_metadata_to_shadow() (*parsley.aspect.metadata._Metadata static method*), 44

set_metadataversion_to_file() (parsley.aspect.metadata._Metadata static method), 44
set_path() (parsley.gui.report.PerformanceData.Loader method), 78
set_retval() (parsley.runtime.runtime.RuntimeData method), 93
set_value() (parsley.runtime.datastorage.StorageDepartment method), 89
set_value() (parsley.runtime.datastorage.SyncVolumeStorageDepartment method), 90
set_value() (parsley.runtime.datastorage.SystemStorageDepartment method), 91
setinfo_current_addtag() (parsley.syncengine.syncruntime.SyncRuntime method), 108
setinfo_removefile() (parsley.syncengine.syncruntime.SyncRuntime method), 108
setinfo_updatefile() (parsley.syncengine.syncruntime.SyncRuntime method), 109
setlastsyncedmtime() (parsley.syncengine.entrylist.CombinedEntryList method), 98
setlastsyncedmtime() (parsley.syncengine.entrylist.EntryList method), 100
setscope() (parsley.runtime.successtracker.SuccessTracker method), 93
setxattrvalue() (parsley.filesystem.abstractfilesystem.Filesystem method), 60
setxattrvalue() (parsley.filesystem.local.LocalFilesystem method), 62
setxattrvalue() (parsley.filesystem.sshfs.SshfsFilesystem method), 63
Severity (class in parsley.logger.logger), 83
shall_skip() (parsley.runtime.successtracker.SuccessTracker method), 94
SHARED (parsley.runtime.datastorage.StorageScope attribute), 89
show() (parsley.gui.app_main.App method), 68
show() (parsley.gui.gtk.volumeapp.AbstractVolumeApp method), 73
shutdown() (parsley.filesystem.abstractfilesystem.Filesystem method), 60
shutdown() (parsley.filesystem.sshfs.SshfsFilesystem method), 63
shutdown() (parsley.filesystem.sshfs.SshfsFilesystem._SshXattrProxy method), 63
SKIP (in module parsley.syncengine.common), 95
skip_master_filesystem_promotion() (parsley.syncengine.syncruntime.SyncEventRuntime method), 104
skip_update() (parsley.syncengine.syncruntime.SyncEventRuntime method), 104
slidetimeinterval (parsley.gui.app_report.timelinepanel.TimelinePanel attribute), 71
spinner (parsley.gui.app_report.performancepanel.PerformancePanel attribute), 70
SshFilesystemGuiCreateHelper (class in parsley.gui.filesystem.sshfs), 72
SshfsFilesystem (class in parsley.filesystem.sshfs), 63
SshfsFilesystem._SshXattrProxy (class in parsley.filesystem.sshfs), 63
SshfsMountPreparation (class in parsley.preparation.sshfsmountpreparation), 86
StorageDepartment (class in parsley.runtime.datastorage), 87
StorageLocation (class in parsley.runtime.datastorage), 89
StorageScope (class in parsley.runtime.datastorage), 89
SUCCESS (parsley.runtime.returnvalue.ReturnValue attribute), 92
successfulcall() (parsley.runtime.successtracker.SuccessTracker method), 94
SuccessTracker (class in parsley.runtime.successtracker), 93
Sync (class in parsley.syncengine.sync), 100
Sync.EventHandler (class in parsley.syncengine.sync), 101
sync_directory() (parsley.syncengine.sync.Sync method), 102
sync_failed (parsley.gui.app_main.runsyncdialog.RunSyncDialog attribute), 68
sync_succeeded (parsley.gui.app_main.runsyncdialog.RunSyncDialog attribute), 68
SYNC_VOLUME (parsley.runtime.datastorage.StorageLocation attribute), 89
SyncConfigurationError, 95
SyncError, 95
SyncEvent (class in parsley.syncengine.common), 95
SyncEventRuntime (class in parsley.syncengine.syncruntime), 103
SyncExecutionError, 97
SyncFilesystemError, 61
SyncLocalFilesystemError, 62

SyncMetadataAspectError, 43
SyncRunLog (*class in parsley.gui.report*), 79
syncruns() (*parsley.gui.report.PerformanceData property*), 79
SyncRuntime (*class in parsley.syncengine.syncruntime*), 104
syncs() (*parsley.config.config.ParsleyConfiguration property*), 54
SyncSshFilesystemError, 63
SyncSshFilesystemRemoteProxyError, 63
SyncTaskConfigItemHandler (*class in parsley.gui.app_main.configitem*), 66
synctasks() (*parsley.gui.report.PerformanceData property*), 79
syncvolumereport_write() (*parsley.aspect.volumesyncreport.VolumeSyncReport method*), 47
SyncVolumeStorageDepartment (*class in parsley.runtime.datastorage*), 89
SYSTEM (*parsley.runtime.datastorage.StorageLocation attribute*), 89
SystemStorageDepartment (*class in parsley.runtime.datastorage*), 90

T

Timeline (*class in parsley.gui.app_report.timelinepanel*), 70
TimelinePanel (*class in parsley.gui.app_report.timelinepanel*), 70
timingdata() (*parsley.gui.report.PerformanceData property*), 79
toolbar (*parsley.gui.app_main.configitem.ConfigItem attribute*), 64
toxml() (*parsley.config.config.ParsleyConfiguration method*), 54
toxml() (*parsley.config.config.ParsleyConfiguration.Aspect method*), 48
toxml() (*parsley.config.config.ParsleyConfiguration.CustomAspect method*), 49
toxml() (*parsley.config.config.ParsleyConfiguration.Filesystem method*), 49
toxml() (*parsley.config.config.ParsleyConfiguration.Include method*), 50
toxml() (*parsley.config.config.ParsleyConfiguration.LogFormatter method*), 50
toxml() (*parsley.config.config.ParsleyConfiguration.Logger method*), 51
toxml() (*parsley.config.config.ParsleyConfiguration.LoggingHandler method*), 51
toxml() (*parsley.config.config.ParsleyConfiguration.Preparation method*), 52
toxml() (*parsley.config.config.ParsleyConfiguration.PythonImport method*), 52
toxml() (*parsley.config.config.ParsleyConfiguration.Sync method*), 53
TrackConflicts (*class in parsley.aspect.conflicts*), 40
trackconflicts_trackskipped_skipconflict() (*parsley.aspect.conflicts.TrackConflicts method*), 40
translate_parsley_portforwarding() (*in module parsley.tools.networking*), 110
translate_path() (*parsley.filesystem.abstractfilesystem.Filesystem method*), 60
translate_to_alternative_endpoint() (*parsley.preparation.sshfsmountpreparation.SshfsMountPreparat static method*), 86
TrashRemove (*class in parsley.aspect.remove*), 46
trashremove_file_link() (*parsley.aspect.remove.TrashRemove method*), 46
tree (*parsley.gui.app_report.performancepanel.PerformancePanel attribute*), 70
try_load_log_from_syncdir() (*in module parsley.gui.report*), 79

U

UnableToGetMountListError, 85
UnableToMountError, 85
UnableToUnmountError, 85
unloadcustomizations() (*parsley.aspect.highlevelcustomization.HighLevelCustomization method*), 43
unlock() (*in module parsley.tools.common*), 110
unsetxattrvalue() (*parsley.filesystem.abstractfilesystem.Filesystem method*), 60
unsetxattrvalue() (*parsley.filesystem.local.LocalFilesystem method*), 62
unsetxattrvalue() (*parsley.filesystem.sshfs.SshfsFilesystem method*), 63
UpdateDir_AfterUpdate (*parsley.syncengine.common.SyncEvent attribute*), 96
UpdateDir_ListDir (*parsley.syncengine.common.SyncEvent attribute*), 96
UpdateDir_Prepare (*parsley.syncengine.common.SyncEvent attribute*), 96
updatefile() (*parsley.syncengine.entrylist.CombinedEntryList method*), 98

updatefile() (*parsley.syncengine.entrylist.EntryList method*), 100
UpdateItem_AfterUpdate (*parsley.syncengine.common.SyncEvent attribute*), 96
UpdateItem_BeforeElectMaster (*parsley.syncengine.common.SyncEvent attribute*), 96
UpdateItem_CheckConflicts (*parsley.syncengine.common.SyncEvent attribute*), 96
UpdateItem_ElectMaster (*parsley.syncengine.common.SyncEvent attribute*), 96
UpdateItem_ResolveConflicts (*parsley.syncengine.common.SyncEvent attribute*), 96
UpdateItem_SkippedDueConflicts (*parsley.syncengine.common.SyncEvent attribute*), 96
UpdateItem_Update_ExistsInMaster (*parsley.syncengine.common.SyncEvent attribute*), 97
UpdateItem_Update_NotExistsInMaster (*parsley.syncengine.common.SyncEvent attribute*), 97
UpdateItem_Update_Prepare (*parsley.syncengine.common.SyncEvent attribute*), 97
UserFeedbackController (*class in parsley.gui.gtk.userfeedback*), 73

V

VALUE_FILENAME (*parsley.runtime.datastorage.StorageDepartment attribute*), 87
value_label (*parsley.gui.app_main.configitem.ParamConfigNode attribute*), 66
version (*parsley.syncengine.benchmark.BenchmarkRun attribute*), 95
VolumePathBreadcrumb (*class in parsley.aspect.volumepathbreadcrumb*), 47
volumepathbreadcrumb_write() (*parsley.aspect.volumepathbreadcrumb.VolumePathBreadcrumb method*), 47
VolumeSyncReport (*class in parsley.aspect.volumesyncreport*), 47

W

write_file() (*parsley.syncengine.syncruntime.SyncRuntime method*), 109
writetofile() (*parsley.filesystem.abstractfilesystem.Filesystem method*), 60
writetofile() (*parsley.filesystem.local.LocalFilesystem method*), 62