ConfigSync Documentation

Release

Martin Simon

January 13, 2017

Contents:

CHAPTER 1

configsync.configsync Module

CHAPTER 2

configsync.configsync_config Module

Module contains objects to manipulate, execute and resolve the ConfigSync configuration.

class `configsync.configsync_config.`**`ConfigSyncConfig`**

Bases: `object`

This class is supposed to manipulate the ConfigSync configuration. It also stores the list of remotelly synchronized files and methods to load/store configuration from/to files.

`restoreConfiguration``()`

Restore stored configuration from a file or create the path to the configuration file and set the default values.

`restoreFileList``()`

Restore file list from remote directory if exists. Otherwise clear the file list as there shouldn't be any files stored. This should be called after ConfigSync.restoreConfiguration() method as the file list should be in the git working directory set in configuration.

`setConfigDefaults``()`

Set default values to configuration if there's no configuration stored yet.

`storeConfiguration``()`

Store configuration into file to make them persistent.

`storeFileList``()`

Store file list to a file on remote directory.

class `configsync.configsync_config.`**`ConfigSyncConfigContainer`**

Bases: `object`

Container of the ConfigSync local configuration. The configuration is not synchronized remotelly and is store under user home directory in .configsync folder.

`addFile``(`*_file*, *_synced*`)`

Add a file into list of synced files. Every entry consists of an original file name and a local file name the remote file is synced with. The 'file name' means a full path in this case.

Parameters

- **`_file`** (*`string`*) – Original file name (remote)
- **`_synced`** (*`string`*) – Local sync target file

`delFile``(`*_file*`)`

Delete a file entry from the list. It occurs when the file is not synchronized any more.

Parameters **`_file`** (*`string`*) – Full path of the synchronized file

existsFile(*_file*)

Check the (remote) file existence in the list of synchronized files.

Parameters **_file** (*string*) – Full path of the expected synchronized file

Returns True if the file exists in the list

Return type bool

getValue(*_file*)

Return a name of the local file which is synchronized with the file given in parameter

Parameters **_file** (*string*) – Full path of the synchronized file

Returns Full path of the local file which is synchronized with the given one

Return type string

class configsync.configsync_config.**ConfigSyncFilesContainer**

Bases: object

Container of file list. This file list is synchornized. The file list consists from trio - file name, file original owner, count of linked sides

addFile(*_file*, *_owner*)

Add a file into a list of remote files. Every file is represented by its name with name of computer of its origin and count of linked sides. The count of linked sides is set to *1*.

Parameters

- **_file** (*string*) – Full name of the synchronized file
- **_owner** (*string*) – Name of the computer of the file's origin

addFileLink(*_file*)

Increase the count of synced sides by *1*.

Parameters **_file** (*string*) – Full name of the synchronized file

delFile(*_file*)

Deete a file entry from the list of synced files. The file should be deleted after the count of synced sides goes bellow *1*.

Parameters **_file** (*string*) – Full name of the synchronized file

delFileLink(*_file*)

Decrease the count of synced sides by *1*. It also checks ify the count goes bellow *1*. It's represented by the return value.

Parameters **_file** (*string*) – Full name of the synchronized file

Returns True if the decreasing makes the count be bellow *1* and the actual file deleting is needed.

Return type bool

existsFile(*_file*)

Check the (remote) file existence in the list of synchronized files.

Parameters **_file** (*string*) – Full name of the synchronized file

CHAPTER 3

configsync.configsync_core Module

Module contains the core of ConfigSync synchronization. It contains all the real functionality of git control and file manipulation. Take a look into implemented methods to get more perspective.

class `configsync.configsync_core.`**`ConfigSyncCore`**`(`*_config=<configsync.configsync_config.ConfigSyncConfig object>*`)`

Bases: `object`

ConfigSync synchronization core.

`createWorkingDirectory``(`*_path*`)`

Create git working directory. In this directory the ConfigSync synchronization git instance is proceed.

Parameters **`_path`** (*`string`*) – Path to the working directory

Returns True if working directory created successfully

Return type bool

`gitAdd``(`*_file*`)`

Add given file to local branch. This is called after every file change.

TODO: The command output should be checked. An error or password or encryption information could be required. The check should be added.

FIXME: The executed command should use native module instead of calling system dependent program.

Parameters **`_file`** (*`string`*) – File basename in working directory

`gitAddFilelist``()`

Add a file list to local branch. This is called after every file change.

TODO: The command output should be checked. An error or password or encryption information could be required. The check should be added.

FIXME: The executed command should use native module instead of calling system dependent program.

`gitClone``(`*_repo*, *_path*`)`

Clone given repository into the given working directory. This method is proceed only once after configuration of ConfigSync is set.

TODO: The command output should be checked. An error or password or encryption information could be required. The check should be added.

TODO: The address should be parsed and checked if this is correct git address.

FIXME: The executed command should use native module instead of calling system dependent program.

Parameters

- **`_repo`** (*`string`*) – Link to the git repository
- **`_path`** (*`string`*) – Path to the working directory

Returns True if clone command passed without any error

Return type bool

`gitCommit``()`

Git commit command. The commit message contains machine name and time stamp.

TODO: The command output should be checked. An error or password or encryption information could be required. The check should be added.

FIXME: The command execution is quite time consuming. It should be done on background.

FIXME: The executed command should use native module instead of calling system dependent program.

`gitPull``()`

Git pull command.

TODO: The command output should be checked. An error or password or encryption information could be required. The check should be added.

FIXME: The command execution is quite time consuming. It should be done on background.

FIXME: The executed command should use native module instead of calling system dependent program.

`gitPush``()`

Git push command.

TODO: The command output should be checked. An error or password or encryption information could be required. The check should be added.

FIXME: The command execution is quite time consuming. It should be done on background.

FIXME: The executed command should use native module instead of calling system dependent program.

`gitRemove``(`*_file*`)`

Remove a file from local branch. This is called after every file remove.

TODO: The command output should be checked. An error or password or encryption information could be required. The check should be added.

FIXME: The executed command should use native module instead of calling system dependent program.

Parameters **`_file`** (*`string`*) – File basename in working directory

`linkFile``(`*_f*, *_s*`)`

Make a hardlink to the file to synchronize to ConfigSync working directory to begin file tracking.

FIXME: The executed command should use native module instead of calling system dependent program.

Parameters

- **`_f`** (*`string`*) – Path to the original file what should be synchronized
- **`_s`** (*`string`*) – Path to the file in ConfigSync working directory

`synchronize``()`

Do the synchronization itself.

TODO: The command output should be checked. An error or password or encryption information could be required. The check should be added.

FIXME: The command execution is quite time consuming. It should be done on background.

`unlinkFile`(*_file*)

Split a hardlink connection between two files to make them independent. This method is called when file should not be synchronized any longer.

FIXME: The executed command should use native module instead of calling system dependent program.

Parameters **`_file`** (*`string`*) – Path to the file in ConfigSync working directory

CHAPTER 4

configsync.configsync_gui Module

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

CHAPTER 6

configsync Package

mod *configsync.configsync* Module

mod *configsync.configsync_config* Module

Module contains objects to manipulate, execute and resolve the ConfigSync configuration.

class `configsync.configsync_config.`**`ConfigSyncConfig`**

This class is supposed to manipulate the ConfigSync configuration. It also stores the list of remotelly synchronized files and methods to load/store configuration from/to files.

`restoreConfiguration``()`

Restore stored configuration from a file or create the path to the configuration file and set the default values.

`restoreFileList``()`

Restore file list from remote directory if exists. Otherwise clear the file list as there shouldn't be any files stored. This should be called after ConfigSync.restoreConfiguration() method as the file list should be in the git working directory set in configuration.

`setConfigDefaults``()`

Set default values to configuration if there's no configuration stored yet.

`storeConfiguration``()`

Store configuration into file to make them persistent.

`storeFileList``()`

Store file list to a file on remote directory.

class `configsync.configsync_config.`**`ConfigSyncConfigContainer`**

Container of the ConfigSync local configuration. The configuration is not synchronized remotelly and is store under user home directory in .configsync folder.

`addFile``(`*_file*, *_synced*`)`

Add a file into list of synced files. Every entry consists of an original file name and a local file name the remote file is synced with. The 'file name' means a full path in this case.

Parameters

- **`_file`** (*`string`*) – Original file name (remote)
- **`_synced`** (*`string`*) – Local sync target file

`delFile``(`*_file*`)`

Delete a file entry from the list. It occurs when the file is not synchronized any more.

Parameters **`_file`** (*`string`*) – Full path of the synchronized file

`existsFile`(*_file*)
Check the (remote) file existence in the list of synchronized files.

Parameters **`_file`** (*`string`*) – Full path of the expected synchronized file

Returns True if the file exists in the list

Return type bool

`getValue`(*_file*)
Return a name of the local file which is synchronized with the file given in parameter

Parameters **`_file`** (*`string`*) – Full path of the synchronized file

Returns Full path of the local file which is synchronized with the given one

Return type string

class `configsync.configsync_config.`**`ConfigSyncFilesContainer`**
Container of file list. This file list is synchornized. The file list consists from trio - file name, file original owner, count of linked sides

`addFile`(*_file*, *_owner*)
Add a file into a list of remote files. Every file is represented by its name with name of computer of its origin and count of linked sides. The count of linked sides is set to *1*.

Parameters

- **`_file`** (*`string`*) – Full name of the synchronized file
- **`_owner`** (*`string`*) – Name of the computer of the file's origin

`addFileLink`(*_file*)
Increase the count of synced sides by *1*.

Parameters **`_file`** (*`string`*) – Full name of the synchronized file

`delFile`(*_file*)
Deete a file entry from the list of synced files. The file should be deleted after the count of synced sides goes bellow *1*.

Parameters **`_file`** (*`string`*) – Full name of the synchronized file

`delFileLink`(*_file*)
Decrease the count of synced sides by *1*. It also checks ify the count goes bellow *1*. It's represented by the return value.

Parameters **`_file`** (*`string`*) – Full name of the synchronized file

Returns True if the decreasing makes the count be bellow *1* and the actual file deleting is needed.

Return type bool

`existsFile`(*_file*)
Check the (remote) file existence in the list of synchronized files.

Parameters **`_file`** (*`string`*) – Full name of the synchronized file

mod *configsync.configsync_core* Module

Module contains the core of ConfigSync synchronization. It contains all the real functionality of git control and file manipulation. Take a look into implemented methods to get more perspective.

class `configsync.configsync_core.`**`ConfigSyncCore`**(*_config=<configsync.configsync_config.ConfigSyncConfig object>*)

ConfigSync synchronization core.

`createWorkingDirectory`(*_path*)

Create git working directory. In this directory the ConfigSync synchronization git instance is proceed.

Parameters **`_path`** (*`string`*) – Path to the working directory

Returns True if working directory created successfully

Return type bool

`gitAdd`(*_file*)

Add given file to local branch. This is called after every file change.

TODO: The command output should be checked. An error or password or encryption information could be required. The check should be added.

FIXME: The executed command should use native module instead of calling system dependent program.

Parameters **`_file`** (*`string`*) – File basename in working directory

`gitAddFilelist`()

Add a file list to local branch. This is called after every file change.

TODO: The command output should be checked. An error or password or encryption information could be required. The check should be added.

FIXME: The executed command should use native module instead of calling system dependent program.

`gitClone`(*_repo*, *_path*)

Clone given repository into the given working directory. This method is proceed only once after configuration of ConfigSync is set.

TODO: The command output should be checked. An error or password or encryption information could be required. The check should be added.

TODO: The address should be parsed and checked if this is correct git address.

FIXME: The executed command should use native module instead of calling system dependent program.

Parameters

- **`_repo`** (*`string`*) – Link to the git repository
- **`_path`** (*`string`*) – Path to the working directory

Returns True if clone command passed without any error

Return type bool

`gitCommit`()

Git commit command. The commit message contains machine name and time stamp.

TODO: The command output should be checked. An error or password or encryption information could be required. The check should be added.

FIXME: The command execution is quite time consuming. It should be done on background.

FIXME: The executed command should use native module instead of calling system dependent program.

`gitPull`()

Git pull command.

TODO: The command output should be checked. An error or password or encryption information could be required. The check should be added.

FIXME: The command execution is quite time consuming. It should be done on background.

FIXME: The executed command should use native module instead of calling system dependent program.

`gitPush``()`

Git push command.

TODO: The command output should be checked. An error or password or encryption information could be required. The check should be added.

FIXME: The command execution is quite time consuming. It should be done on background.

FIXME: The executed command should use native module instead of calling system dependent program.

`gitRemove``(`*_file*`)`

Remove a file from local branch. This is called after every file remove.

TODO: The command output should be checked. An error or password or encryption information could be required. The check should be added.

FIXME: The executed command should use native module instead of calling system dependent program.

Parameters **`_file`** (*`string`*) – File basename in working directory

`linkFile``(`*_f*, *_s*`)`

Make a hardlink to the file to synchronize to ConfigSync working directory to begin file tracking.

FIXME: The executed command should use native module instead of calling system dependent program.

Parameters

- **`_f`** (*`string`*) – Path to the original file what should be synchronized
- **`_s`** (*`string`*) – Path to the file in ConfigSync working directory

`synchronize``()`

Do the synchronization itself.

TODO: The command output should be checked. An error or password or encryption information could be required. The check should be added.

FIXME: The command execution is quite time consuming. It should be done on background.

`unlinkFile``(`*_file*`)`

Split a hardlink connection between two files to make them independent. This method is called when file should not be synchronized any longer.

FIXME: The executed command should use native module instead of calling system dependent program.

Parameters **`_file`** (*`string`*) – Path to the file in ConfigSync working directory

mod *configsync.configsync_gui* Module

Python Module Index

Index