
RAFCON API

Release 2.1.2

Rico Belder, Sebastian Brunner, Franz Steinmetz

Mar 25, 2024

TABLE OF CONTENTS

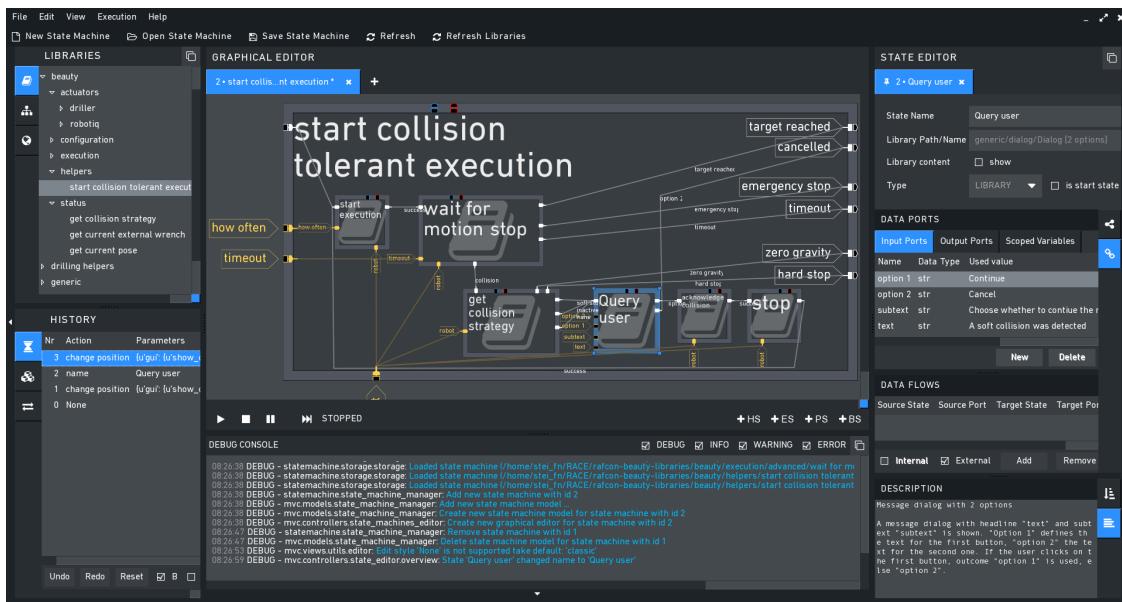
1 Concepts	3
2 Changelog	7
3 List of Features	47
4 Breaking Changes	51
5 Tutorials	53
6 Building the documentation	65
7 Configuration	67
8 Auto Backup	81
9 GUI Guide	83
10 Plugins	87
11 FAQ	91
12 Contributing: Developer's Guide	99
13 RAFCON API: The rafcon package	111
14 Indices and tables	281
Python Module Index	283
Index	287

RAFCON

Develop your robotic tasks using an intuitive graphical user interface

RAFCON (**R**M**C** advanced Flow **C**ontrol) uses hierarchical state machines, featuring concurrent state execution, to represent robot programs. It ships with a graphical user interface supporting the creation of state machines and contains IDE like debugging mechanisms. Alternatively, state machines can programmatically be generated using RAFCON's API.

RAFCON is licensed under the Eclipse Public License - v 1.0 (EPL).



Have a look at our [website](#) for an introduction about installing RAFCON.

CONCEPTS

A state-machine in *RAFCON* consists of states connected with transitions. In addition to this logical flow, RAFCON introduces what is called a data flow. This allows data to be directly shared between states.

1.1 States

A state does not represent a state of the system, but a state in task flow. Usually a state executes some action or calculates some result and finally return an outcome. States can have an arbitrary number of input ports and output ports. There are different types of states:

The simplest type of state is called **Execution state**. An execution state includes an `execute` function that is run when the corresponding state is entered. The parameters of the function are references to the input and output data as well as to the global variable manager. The return value of the `execute` function determines the outcome of a state.

A **Hierarchy state** is one kind of container state. It groups one or more child states (which can be containers themselves). In addition to the input and output ports, hierarchy states can have scoped variables. Hierarchy states do not have an `execute` function, they are just means to encapsulate other states.

Concurrency states are also container states. They allow for the parallel execution of two or more states. All direct children are executed in parallel in different threads. Parallel executing states cannot exchange data using data flows, but have to use global variables with the global variable manager. Global variables are thread safe. There are two types of concurrency to be distinguished:

Barrier concurrency states wait for all parallel executing states to finish. After all child states are finished a special **Decider** state is called. This state decides upon the final outcome of the barrier state itself.

Preemptive concurrency states can have several outcomes. The first parallel executing state that finishes determines the outcome and forces all other states to preempt.

A special type of states are **Library states**. These states cannot be edited within a state-machine, but are predefined. A library state is a state-machine itself and can internally consist of arbitrary states and hierarchies (even further libraries). Every state-machine can be converted to a library and thus be reused in other projects. Libraries can be parameterized like any other state with input ports and return values/results on output ports.

1.2 Outcomes

Outcomes can be described as “exit points” of states. Every state has at least two outcomes, “aborted” and “preempted”. In addition to these special outcomes, every state can have an arbitrary number of additional outcomes with individual names (“success”, “ $>=0$ ”, “yes”, …). Each outcome has an id. For “aborted” it is -1 and for “preempted” it is -2. Further outcomes have increasing numbers starting from 0. Depending on the type of the state, the outcome on which the state is left is determined in different ways:

- For execution states, the user determines the outcome by returning its id at the end of the `execute` function. If the returned id is not existing, this is treated as an error and the state is left on the “aborted” outcome. Errors and exceptions in general force a state to be left on the “aborted” outcome.
- For hierarchy states, the child states together with the transitions determine the outcome. As soon as during the execution of a hierarchy state a transition leads to an outcome of the hierarchy state, the state is left on this outcome. However, if an exception or error occurs in a child state, which is not caught (by connecting the “aborted” outcome of the erroneous state), the hierarchy state is left also on the “aborted” outcome. By this, errors and exception climb up the hierarchy, until an “aborted” outcome is connected or the root state is reached (and the program stops).
- The situation for preemptive concurrency states is similar to that of hierarchy states. A triggered transition to an outcome causes the state to be left on that outcome. In addition, all parallel states are left (from the innermost running state upwards) on the “preempted” outcomes. This allows states to ensure a safe and consistent state when being forced to stop.
- Barrier concurrency states are special in that the Decider is responsible for the final outcome. This state decides based on the outcomes of the parallel child states and the output data of the child states the final outcome of the barrier state.

1.3 Transitions

Transitions connect outcomes and states (or other outcomes). They do not have a name and or any further data. Every outcome should be connected with a transition, which is connected to another state (there is only one “entry point” per state, where several transition can be connected to). In container states (hierarchy or concurrency states) transitions can also be connected from a outcome of child state to another outcome of the parent state. This causes the execution flow to leave the hierarchy at that point. There may only be one transition per outcome.

1.4 Data ports

Data ports are the equivalent to outcomes, but on the data flow level (in contrast to the logical flow). Data ports have a name, a type and a default value. There are two types of data ports, input data ports and output data ports (from hereon called inputs and outputs). Every state can have an arbitrary number of inputs and outputs, which are used to pass data in and out.

Inputs can be compared to parameters of a function. In our case, the function is the state. A state defines which data it needs to calculate its output or to execute some action. The data fed to input ports is passed to the `execute` function of execution states. It is the first parameter of type `dict` and the names of the inputs as keys. If no data flow is connected to an input, then the default value is used as its value.

Accordingly, outputs are the return values of states. Similar to Python functions, states cannot only return one value, but arbitrarily many (or none). The outputs are also passed to the `execute` function of execution states. It is the second parameter, also of type `dict`. The function can set the value of each output by assigning a value to the according dictionary entry. If no value is set, the default value is used for the further execution.

It is important to note that the input values used are passed by value and this value is created as deep copy either from the value coming from the data flow or the default value. As a consequence, complex values (e. g. dictionaries) calculated by one state, which are fed to two different states, can be modified in one state without the other state seeing this modification. One state can be executed several times when being in a loop and always gets new input data.

The data types of individual data ports can be of standard python built-in data types (e.g. int, float, long, complex, str, unicode, tuple, list, dict, bool). They can also be of types defined by third-party packages, as long as the packages lie in the PYTHONPATH environment variable (e.g. numpy.ndarray).

1.5 Scoped variables

Scoped variables only exist in container states and have a name, type and default value (just like data ports). They can be seen as kind of variable or data port, from which every child state can read from and write to. Thus, they can for example be used as data storage for states being executed several times (using loops).

1.6 Data flows

Data flows are for data ports (and scoped variables) what transitions are for outcomes. Just as transitions, they neither have a name nor hold any further data. They define the flow of data, typically from outputs (sources) to inputs (sinks). However, it is not as simple as that. In the case of container states, a data flow can go from the input of the container state to the input of a child state (feeding data down in the hierarchy). Similarly, data flows can go from the output of a child to the output of its container state (feeding data out/up the hierarchy).

In addition, inputs can receive (read) data from scoped variables and outputs can pass data (overwrite) to scoped variables. A container input can write to a scoped variable as well as the scoped variable can write to an output of its container.

A port (input, output, scoped variable) can serve both as source and of sink of data flows for arbitrary many other ports.

1.7 Global Variable Manager

The Global Variable Manager (GVM) allows to store values globally. The GVM is thread-safe, thus you can access variables from in parallel running states. The GVM is intended for variables, which are needed in many states within different containers, such as constants, global parameters, etc. or for global objects, such as a middleware client. You should not abuse the GVM as a general replacement for data ports and data flows. Actually, modelling parameters via data flows has huge advantages: It enables easy encapsulation and reuse of your states, makes your state machines easy to understand and can be nicely logged in the execution history (which also can be analyzed after execution).

The `execute` function of Execution States retrieves a reference to the GVM as its third parameter after `self`. Variables are set using `set_variable(self, key, value, per_reference=False, access_key=None)`. Parameter `key` is the name of the variable, `value` is the (new) value. If the variable is not existing, it is created, otherwise the value is overwritten. If you only want a reference to be stored, set `per_reference` to True, otherwise a deep copy is created. If the variable is locked, you have to specify the `access_key` to temporary unlock it, otherwise a `RuntimeError` is raised.

To read the value of a variable stored in the GVM, use `get_variable(self, key, per_reference=False, access_key=None, default=None)`. The `key` is again the name of the variable. If it is not existing, the value specified by the `default` parameter is returned. If you only want a reference to the value, set `per_reference` to True. This is only possible for variables, whose value was stored by reference. Again a `RuntimeError` is raised in case of a failure. Specify the `access_key`, in case the variable is locked, otherwise a `RuntimeError` is raised.

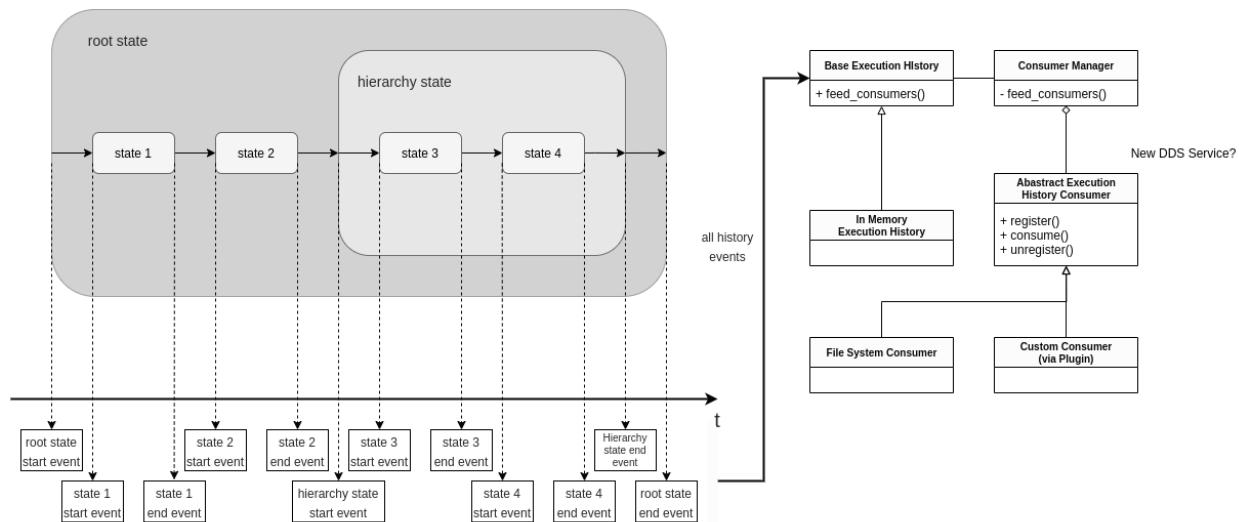
Variables can be locked to prevent access from other states. To do so, call `lock_variable(self, key)` and specify the variable name with `key`. The access key is returned, which is needed to unlock the variable again with `unlock_variable(self, key, access_key)`.

Often, you will want to pass the value of a variable stored in the GVM to an input port. For this, a short-hand method was introduced. All you have to do is setting the default value of the input port to `$key`, where `key` is the name of the variable. If the variable is not existing, the port value is set to `None`.

You can see the current variables of the GVM and their values in the left-hand side of the GUI. There you can also create new variables. However, variables are not stored when saving state-machines. If you want to have variables loaded with the state-machine, you have to create those variables in an initial execution state.

1.8 Execution History

The general idea of the execution history is shown in the following figure:



During execution, a start and an end event is created for each state. Subsequently, the events are forwarded via the `ExecutionHistory` (in case the `IN_MEMORY_EXECUTION_HISTORY_ENABLE` is enabled, the `InMemoryExecutionHistory` is used) to the `ConsumerManager`. The `ConsumerManager` distributes the events to different Consumers. RAFCON ships with a default Consumer called the `FileSystemConsumer`. If `FILE_SYSTEM_EXECUTION_HISTORY_ENABLE` is enabled the `FileSystemConsumer` will write all history on disk into a shelfe file. The `execution_log_viewer.py` can then be used to analyze the execution history logs after un-time.

It is straightforward to create other consumers (e.g. for logging the execution-history via a middleware). Therefore, a plugin can be created that just works similar to the `FileSystemConsumer`. For writing a plugin, only the “`pre_init`” and the “`register_execution_history_consumer`” hook has to be created. The latter one has to care about that the consumer is registered at the `ExecutionHistoryConsumerManager`.

CHANGELOG

Information about *RAFCON* changes in each release will be published here. More details can be found in the [GIT commit log](#).

2.1 2.1.2

- **Bug fixes:**

- Fixed bug for logic connectors
- Added logo to about dialog

- **Miscellaneous:**

- Updated pdm.lock
- Added minimal rafcon core tutorial in docs
- Updated formatting and deprecation for docs
- Updated citation formatting

2.2 2.1.1

- **Bug fixes:**

- Updated deprecated dependency to jsonconversion
- Added new test state machine for backward compatibility testing

2.3 2.1.0

- **Features:**

- Added option to disable popups in the config file
- Display warning when saving new state machine (and overwriting) in already existing folder

- **Bug fixes:**

- Fixing segmentation fault when changing state type in gui via state editor
- Fixing decider node is not preempted in concurrency state

- Fixed warnings and bugs in unit tests
- More minor bugfixes
- **Miscellaneous:**
 - Checked dependencies and deprecations for libraries
 - Updated shebang versions to python3
 - Removed some warnings resulting from old python conventions

2.4 2.0.0

- **Features:**
 - Switch from setup.py to pyproject.toml and pdm for the python package management
 - Add bump2version to avoid human errors when updating the rafcon version
 - Added auto-layout functionality (as a first version)

2.5 1.3.0

- **Features:**
 - Add possibility to only release rafcon-core

2.6 1.2.1

- **Features:**
 - Add __main__.py

2.7 1.2.0

- **Features:**
 - Support python 3.10

2.8 1.1.1

- **Bug Fixes:**
 - Fix a few GUI bugs

2.9 1.1.0

- **Features:**

- Add skip & skip all buttons in the dialog of the broken libraries during loading a broken session
- Set the default directory of the dialog during saving a state machine in accordance with the chosen library in the library tree
- Create the data flows & data ports automatically in the nested states
- Create the data port automatically when the other state does not have one during connecting the data flows
- Support waypoints for data flows
- Custom background color for states

- **Bug Fixes:**

- Fix many minor GUI bugs

2.10 1.0.1

- **Bug Fixes:**

- Fix the default primary font name

2.11 1.0.0

- **Features:**

- Not supporting Python 2 anymore
- Run this state
- Only run this state
- Add singleton pattern
- Add new hooks before and after running each state
- Add new memory profiling test to assert the memory leak during running sequential & concurrency state machines
- Update gaphas to 2.1.2
- Update libsass to the latest version of dart sass
- Replace gtkmvc3 with two separated mvc and observer patterns
- Run this state

- **Bug Fixes:**

- Fix GUI freezing during keeping undo/redo shortcuts

- **Miscellaneous:**

- Remove last update field to improve versioning control
- Remove a big amount of the dead codes and comments

2.12 0.15.4

- **Bug Fixes:**
 - Support custom design

2.13 0.15.3

- **Bug Fixes:**
 - Fix bug in LoggingView, which freezes RAFCON

2.14 0.15.2

- **Bug Fixes:**
 - Make operations on the logging console thread-safe
 - Define a new GUI config called ‘MAX_LOGGING_BUFFER_LINES’ that determines the maximum lines of the logging buffer. If the number of lines exceeds the config value, the old value will be deleted automatically via clipping.

2.15 0.15.1

- **Bug Fixes:**
 - Call ‘show_notification’ via ‘idle_add’

2.16 0.15.0

- **Features:**
 - Libraries can now be renamed and relocated. This includes single libraries, library folders and library root keys
 - Ctrl+F can be used to search for states
 - Missing libraries are supported better. In case a library cannot be found, the transitions and data-flows are preserved and added to the dummy-state, which is inserted instead of the library. Furthermore, the dummy-state has the same position and size as the old library state.
 - New execution-history structure: Define specific consumers for in-memory-execution-history and file-system execution history. Furthermore, another hook was defined such that RAFCON plugins can be used to define further consumers. Watch out: the config values for controlling the execution history changed

2.17 0.14.11

- **Features:**
 - Add search bar for lookup through state machine libraries
 - Add find usage for finding the usages of state machine libraries
- **Bug Fixes:**
 - Fix handling of library interface change

2.18 0.14.10

- **Features:**
 - Add new config (RAISE_ERROR_ON_MISSING_LIBRARY_STATES) to make Rafcon raise error when loading

2.19 0.14.9

- **Features:**
 - add states for execution control

2.20 0.14.8

- **Bug Fixes:**
 - Fix py2 support

2.21 0.14.7

- **Features:**
 - increase test coverage
 - add gitlab runners support
 - differentiate between py3 and py2 dependencies in setup.py
 - differentiate between EXECUTION_LOG_ENABLE and EXECUTION_LOG_TO_FILESYSTEM_ENABLE config options i.e. keep memory footprint of RAFCON constant
 - add memory leak test
 - Fix race condition in ‘call_gui_callback’

2.22 0.14.6

- **Miscellaneous:**
 - fix buggy pypi upload

2.23 0.14.5

- **Bug Fixes:**
 - execution log viewer now works via released script in bin folder

2.24 0.14.4

- **Features:**
 - Issue #290 paste state at current mouse position (both via context menu and shortcut) @CSuerig
 - add state at context menu position when using context menu to add states @CSuerig

2.25 0.14.3

Maintenance release.

2.26 0.14.2

- Features:
 - Replace `SCRIPT_COMPILE_ON_FILESYSTEM_LOAD` in favor of `SCRIPT_RECOMPILATION_ON_STATE_EXECUTION`. See the documentation of the configuration for details.
- Bug Fixes:
 - Issue #28 Setting of external editor via dialog does not work
 - Issue #790 `gui_config.yaml` not saved anymore automatically
 - Make tests run with `pytest-mock>=1.11.2`
 - Add compatibility with `pylint>=2.4`
 - Positions of panes should be restored correctly
 - Fix several deprecation warnings
- Miscellaneous:
 - do not test Python 3.4 on Jenkins
 - Coverage test only on `develop` and `master` branch
 - prepare for new `yaml_configuration` release

2.27 0.14.1

- Bug Fixes:
 - [Issue #774](#) python setup.py build_sass not working
 - [Issue #26](#) python3's “`__pycache__`” folder chrashes loading of examples

2.28 0.14.0

- Features:
 - new notification bar, informing about important log entries (configurable), fixes [Issue #288](#)
 - Fullscreen mode: optionally show toolbar (`FULLSCREEN_SHOW_TOOLBAR` option), show notifications
- Improvements:
 - most [PyGTK]DeprecationWarnings are fixed
 - graphical editor: minor performance optimizations
 - specify separators for JSON files: Python 3.4 no longer changes the whitespaces in state machine files
 - override builtins string in JSON files: state machine files generated by Python 2 and 3 are now fully identical
 - code coverage report in Jenkins
 - shows RAFCON log messages during installation
 - parallel test runs on Jenkins
 - [Issue #21](#) Do not store semantic data if not available
 - [Issue #665](#) Keep root state position when collapsing left sidebar
 - better defaults:
 - * root state is named “root state”, further states “[state type] [states counter]”
 - * script of `ExecutionStates` uses more RAFCON features (`preemptive_wait`, return outcome name)
 - * name of states uses full width of state
 - provide RAFCON wheel file
 - make installation more robust, especially against missing font files
 - simplify installation process
 - clear separation in handling of `data_files` and `package_files`
 - create translation files automatically when building dist packages
 - refactored many parts of modification history
- Bug Fixes:
 - [Issue #20](#) program icon in task bar missing since version 0.13.x
 - [Issue #665](#) state type dropdown menu prevents state editor widget to shrink
 - [Issue #694](#) json library in python 3.6 writes one-line json files
 - [Issue #721](#) Correct execution history logging
 - [Issue #726](#) State with self-transition cannot be substituted

- [Issue #727](#) Sticky-Flag in States-Editor can cause crash if state type change is performed
- [Issue #755](#) Positions of outcomes are not always updated
- fixes bug of “locked” global variable during multithreading access
- use a safe loader for GUI config file
- fix handling of symlinks in LibraryManager
- better support of virtual envs
- Changes:
 - drop support for BuildBot
 - Jenkinsfile: tests are now also run under Python 3.6
- Miscellaneous:
 - new gui fixture for simplifying GUI tests
 - refactor GUI tests using the gui fixture
 - documentation on how to write tests and how to use gui fixture

2.28.1 Patch releases 0.13.*

2.29 0.13.8

- Improvements:
 - use with statement instead af acquire/release
 - dedicated ‘unstable’ marks for python 2.7 and 3.x; these marks can be used to filter out tests
 - use Python warning module with custom RAFCONDeprecationWarning for deprecated usages
 - the documentation can again be build on Read The Docs (at least the build of the API docs was corrupt since v0.13)
 - tooltip of library tree include root state description text of libraries
 - Jenkins integration
 - test adaptions so that they can be parallelized
 - added *seqm.yaml* for tracking software engineering quality management (SEQM) requirements (DLR internal)
- Bug Fixes:
 - [Issue #12](#) Error when switching from python2 to python3
 - [Issue #18](#) State machines with library states cannot be opened if show flag is set to True
 - [Issue #683](#) rafcon can now be closed properly via signal
 - [Issue #712](#) Paste of Port into selected state is not possible
 - [Issue #711](#) Gaphas does not allow data flows from one state to itself
 - [Issue #717](#) States that have data-flows from its output to its input crash gahpas while state type change
 - fix broken links in documentation
 - use correct version and year in documentation

- Changes: - pyyaml is not a dependency anymore, as it is now a dependency of yaml_configuration

2.30 0.13.7

- Improvements:
 - add tox integration
 - * run tests under Python interpreters 2.7, 3.4, 3.5, 3.6, 3.7
 - * run tests with coverage
 - * build documentation and check links
 - * check sdist
 - optimize setup_requires in setup.py (faster installation)
 - mark unreliable tests as unstable
 - define timeouts for all tests
- Bug Fixes:
 - [Issue #689](#) rafcon cannot run without numpy
 - [Issue #679](#) error message when connecting data flow
 - fix severe threading bug in call_gui_callback, which could lead to a complete freeze of a state machine

2.31 0.13.6

- Features:
 - add ExecutionTicker to see activity of state machine with high hierarchy depth
- Improvements:
 - changing states (adding or removing) during step mode works now
- Bug Fixes:
 - [Issue #678](#) script validation does not work
 - [Issue #663](#) cannot rename connected data port of type object
 - [Issue #684](#) test_simple_execution_model_and_core_destruct_with_gui fails when running core & gui tests in a row
 - fix pause and step mode behavior
 - installation of fonts under Python 3
 - various test fixed for Python 3

2.32 0.13.5

- Bug Fixes:
 - Continue installation of none-existing fonts in case that one font was already installed

2.33 0.13.4

- Bug Fixes:
 - Fix installation of not-existing fonts
 - Issue #660 tab of executed state machine stays green
 - Issue #667 dialog “saving state as library” not working properly
 - Issue #664 cleaning of execution history does not work
 - Issue #668 adding a state as template screws up meta data
 - Fix rescaling factor**2 if adding libraries as template
 - Issue #631 Cut of multiple states creates various problems
- Changes:
 - Increase any MAX_VISIBLE_LIBRARY_HIERARCHY value to be minimal 2 -> for performance the aim is to allow lower values again

2.34 0.13.3

- Changes:
 - Release correct style files

2.35 0.13.2

- Features:
 - The right click menu of library state can be used to select and focus respective library tree element
- Bug Fixes:
 - Issue #658 crash in load_state_machine
 - run correct command for updating font cache
- Changes:
 - Replaced font “DIN Next LT Pro” by “Source Sans Pro”

2.36 0.13.1

- Bug Fixes: Fix installation

2.37 0.13.0

This is a shiny new minor release of RAFCON. Finally, Python 3 (≥ 3.4) is supported, while Python 2.7 can still be used, thanks to the `future` packet. With this, we also ported the GUI from GTK+ 2 to GTK+ 3, allowing for better styling. Of course, there are many more improvements and bug fixes:

- Features:
 - RAFCON is now compatible to Python 3
 - GTK+ 2 to GTK+ 3 port of the RAFCON GUI
 - Better styling including a HeaderBar
 - Alternative light theme! (GUI config option `THEME_DARK_VARIANT`)
- Improvements:
 - [Issue #117](#) Make GUI resizeable on all edges and corners
 - [Issue #610](#) Provide CITATION.cff to make software citable
 - [Issue #619](#) Provide and install *.desktop file
 - [Issue #621](#) Provide full license text
 - [Issue #636](#) No exception when closing RAFCON and a state machine is still running
 - [Issue #637](#) No exception when closing a state machine tab, when it still runs
 - [Issue #640](#) Backward compatibility test runs with various python versions now
 - [Issue #646](#) Library roots can be added and removed inside the library tree
 - The installation should now work from a blank virtualenv
 - The documentation about the release steps has been extended
- Bug Fixes:
 - [Issue #596](#) External editor does not remember the handed command and also does not lock the embedded editor
 - [Issue #617](#) Invalid DataFlow by DataFlowWidget
 - [Issue #618](#) semantic data strings get scrambled/obfuscated in execution history log fixed by pull request [Issue #626](#) fix(execution_log): unpickle semantic data
 - [Issue #624](#) Debug console: cursor is not positioned at the point were it is clicked on
 - [Issue #627](#) Generic library state machines need Gtk2 to gtk3 conversion
 - [Issue #638](#) Exiting Fullscreen mode hides the graphical editor
 - [Issue #644](#) “Substitute state as template” creates problems if not all models are recursive created
- Changes:
 - Redundant libraries are marked as deprecated
 - No more “+”-icon next to state machine tabs to add a new state machine (related to [Issue #639](#))

- Remove old OpenGL GraphicalEditor
- Remove deprecated entry points `rafcon_start` and `rafcon_start_gui`

2.37.1 Patch releases 0.12.*

2.38 0.12.25

- Improvements:
 - A `DataPort` with data type `object` can now be connected to any other `DataPort` ([Issue #422](#), [Issue #525](#))
 - [Issue #602](#) Hide menu entries without function
 - Handle exceptions of the OpenGL graphical editor gracefully => do not depend on `gtkglext`
- Bug Fixes:
 - no more `GtkWarning` in stdotu
 - [GitHub Issue #4](#) GTK theme does not exist

2.39 0.12.24

- Improvements:
 - Update documentation regarding installation
- Bug Fixes:
 - Installation of mo-files (for language support) works

2.40 0.12.23

- Improvements:
 - Update documentation regarding installation
 - Update rafcon dependencies in `setup.py`
- Bug Fixes:
 - API: `AttributeError` when passing `DeciderState` to constructor of `BarrierConcurrencyState`
 - Installation of mo-files (for language support) works

2.41 0.12.22

- Features:
 - [Issue #581](#) Utility shortcuts to add transitions from selected state to parent default outcome and sibling states
- Improvements:
 - redraw graphical editor if connections are removed
 - extend German RAFCON translation
 - extend Developer's Guide by how-to on translating RAFCON
 - API: `add_state` is adapting the passed `state.state_id` automatically in case of conflicts instead of raising an `AttributeError`
- Bug Fixes:
 - [Issue #455](#) Proportional resizing states now works properly
 - [Issue #538](#) Many error outputs when changing `MAX_VISIBLE_LIBRARY_HIERARCHY`
 - [Issue #541](#) Where are the magnet lines gone?
 - [Issue #551](#) Prevent RAFCON from restarting if installation of fonts fails
 - [Issue #571](#) Wrong rendering of scoped variables
 - [Issue #580](#) update font installation
 - [Issue #584](#) Opening a external source editor fails for a never set active state machine id
 - [Issue #586](#) Ungroup of a state with data flows in between of its child states twice in the same hierarchy creates corrupt state machine or fails
 - stepping works inside library and concurrency states
 - [Issue #589](#) decider state can be deleted
 - make i18n work

2.42 0.12.21

- Features: - new save state machine as menu item for root state right click menu to offer direct ‘save as library’ operations
- Improvements:
 - [Issue #579](#) Integrate external execution log viewer
- Bug Fixes:
 - [Issue #574](#) Group fails if it includes data flows between the grouped states or scoped variables

2.43 0.12.20

- Features:
 - maintenance release

2.44 0.12.19

- Bug Fixes:
 - fix setup.py, sdist now working on pypi

2.45 0.12.18

- Features:
 - new shortcut open library state separately as state machine by default on ‘Shift+Ctrl+Space’ (shortcut works for multiple states, too)
- Improvements:
 - Provides proper PyCharm config files (in the *.idea* folder)
 - update menu item labels
 - updated rst documentation
- Bug Fixes:
 - recent opened state machine list no more miss paths
 - [Issue #550](#) Gaphas cairo.Error: invalid value (typically too big) for the size of the input (surface, pattern, etc.)
 - [Issue #564](#) Zoom onto mouse position
 - handle config option *ZOOM_WITH_CTRL* properly

2.46 0.12.17

- Improvements:
 - example state machines and generic libraries get now installed via pypi

2.47 0.12.16

- Improvements:
 - default config file extended

2.48 0.12.15

- Improvements:
 - PYTHONUSERBASE added to search path list for gtk style files

2.49 0.12.14

- Improvements:
 - library_manager: increase performance of loading libraries by caching a list of all loaded libraries
 - gaphas editor: use new meta data hash method to speed up loading time

2.50 0.12.13

- Improvements:
 - the column headers of state machine tree now can be used to sort the items according state name, ID or type
 - more user friendly interface for tree and list view widgets e.g. data ports, outcomes and semantic data -> scrollbar adjustment and selections are moving much less and try to stay in the front of respective widget
 - correct tab motion to be more accurate
 - execution_history widget shows more visible chars per data port

2.51 0.12.12

- Improvements:
 - [Issue #530](#) automatically focus and adapt position of root state for fresh initiated state machines issue title was “Root state out of focus and badly positioned”
 - [Issue #543](#) Changing default option for library name while saving -> for the default folder name white space are replaced with underscores and all is lower case
 - also default library state name is now the folder name with replaced underscores with white spaces
- Bug Fixes:
 - [Issue #527](#) RAFCON GUI loops while startup if HOME environment variable is not defined -> a error message pointing on respective missing environment variable is added
 - [Issue #539](#) grouping of states outcome transitions are not fully recovers (now bug is covered by test)
 - [Issue #515](#) source editor does not show end of lines (finally)

2.52 0.12.11

- Improvements:
 - [Issue #529](#) accelerate the follow mode switch for many logger messages
 - dynamic insertion of states during state execution is working and tested
 - secure dynamic modification of state machines while runtime by test created in pull request [Issue #535](#)
Dynamic insertion of states during execution
- Bug Fixes:
 - [Issue #515](#) source editor does not show end of lines (partly)
 - [Issue #533](#) States inside library states cannot be selected
 - [Issue #528](#) execution history destruction does not lead to max recursion depth

2.53 0.12.10

- Features:
 - [Issue #520](#) Debug Console keeps track of last logger message if the follow mode is enabled
- Improvements:
 - in pull request [Issue #523](#) refactoring of debug console for more intuitive and robust behavior e.g. persistent cursor position
 - [Issue #516](#) source editor does not show line of cursor after apply if the script is big
- Bug Fixes:
 - [Issue #519](#) rafcon freezes while opening a state machine - solved in pull request [Issue #524](#) history elements hold direct state reference
 - [Issue #514](#) text in entry widget of port not visible during editing (arrow key press left-right helps) - the issue was not fully resolved but improved

2.54 0.12.9

- Improvements:
 - container state API can adjust output_data by new method write_output_data
 - more robust execution history tree
 - performance improvement by deleting gaphas views at once for recursive state destruction's
- Bug Fixes:
 - [Issue #521](#) Strange gaphas logs during deletion of a state
 - fix gaphas exceptions if state machine selection holds elements which gaphas has not drawn

2.55 0.12.8

- Feature:
 - start RAFCON with *rafcon* instead of *rafcon_start_gui* or *rafcon_core* instead of *rafcon_start* (old commands are still working)
- Improvements:
 - buttons to forcefully lock or unlock a global variable
 - global variable manager logger messages got new failure warning messages
 - copy/paste for semantic data elements
 - new config value SHOW_PATH_NAMES_IN_EXECUTION_HISTORY
 - make library path in state editor overview selectable
- Bug Fixes:
 - [Issue #503](#) scoped variable looks weird
 - [Issue #505](#) clean up profiler flag in config
 - [Issue #506](#) root state input ports leave ugly stripes behind
 - [Issue #501](#) transition is not selectable if it is drawn over state
 - [Issue #512](#) execution of second state machine cause freeze of stop on previous state machine was not successful
 - [Issue #514](#) text in entry widget of port not visible during editing
 - fix state machine tree remove library state
 - no deadlocks when locking a global variable two times
 - [Issue #502](#) changing data ports not possible
 - fix state element weakref parent assignment in case of tolerating a invalid data flow

2.56 0.12.7

- Improvements:
 - updated documentation
 - use verbose logging level instead of prints for modification history debug prints

2.57 0.12.6

- Feature:
 - tests folder is now released as well
- Bug Fixes:
 - fix open-gl support for show-content to support fast state machine exploration (also into all leaf-states by zoom)
 - library state can be removed also when those are showing content

2.58 0.12.5

- Feature
 - new log level “VERBOSE”, intended for development purposes
 - state machines can now be baked (a snapshot of the state machine with all libraries can be saved)
 - Graphviz can now be used to debug gtkmvc notifications and signals
- Improvements:
 - Gtk priority of logging output to the console view is now customizable via the gui_config
 - better plugin support of changes to the state-editor tabs
 - gaphas combines now complex meta data actions in one meta data changed signal -> one undo/redo-Action
- Bug Fixes:
 - Issue #484 label handles are hard to grasp
 - Issue #486 Gaphas is not emitting meta data signal if NameView is moved
 - quick fix for not working “state type change” in combination with library states (which was based on respective object destruction while those operations) -> will be fully solved in Issue #493
 - quick fix for not set or too late set of active state machine id -> will be fully solved in Issue #495
 - fix meta data for undo/redo of add object operations
 - fix exception handling, causing issues with the graphical editor when invalid connection were created
 - When hovering the menu bar, an exception was printed

2.59 0.12.4

- Improvements:
 - Provide a *PULL_REQUEST_TEMPLATE* for pull requests opened in GitHub
 - Optimize updates/redrawing of graphical editor
- Bug Fixes:
 - Issue #414 state machines with libraries cannot be closed

2.60 0.12.3

- Feature
 - The env variable RAFCON_START_MINIMIZED allows to start RAFCON minimized, which is helpful when running the tests
- Improvements:
 - Issue #414 Memory optimizations: The memory usage should no longer increase over time, as unused objects are now freed
 - A new/extended test verifies the correct destruction of removed elements
 - Optimize NameView font size calculations, noticeable during zooming

- ports outside of the visible view are no longer drawn, which increases the performance, especially while zooming in large state machines
 - Hash calculations of state machines
 - Placement of NameView
 - drawing of connections, ports and labels, especially when deeply nested
 - [Issue #469](#) unit test refactorings
- Bug Fixes:
 - [Issue #459](#) execution_log utils; backward compatibility missing and [Issue #458](#) ReturnItem
 - [Issue #454](#) group/ungroup is not preserving meta data recursively
 - [Issue #452](#) Session restore, gaphas and extended controller causes exception when closing RAFCON
 - [Issue #450](#) Names of states inside a library become smaller
 - [Issue #447](#) Hashes of state machine in storage different then the reopened state machine after saving it
 - [Issue #449](#) ports (of transitions or data flows) cannot be moved
 - [Issue #471](#) selection of states in hierarchies ≥ 5 not possible

2.61 0.12.2

- New Features:
 - Fix logging for library state execution
- Improvements:
 - Improve execution logging (semantic data is supported now)
 - [Issue #445](#) Tests need to ensure correct import order for GUI singletons
- Bug Fixes:
 - [Issue #446](#) “show content” leads to sm marked as modified

2.62 0.12.1

- New Features:
 - Semantic data editor supports external editor
 - Transparency of library states improved when content is shown
- Improvements:
 - [Issue #415](#) Increase visibility of library content
- Bug Fixes:
 - [Issue #378](#) Editing default values does not work sometimes

2.63 0.12.0

- New Features:
 - Semantic meta data editor and storage for every state
 - [Issue #411](#) Allow outputting data from preempted states
- Bug Fixes:
 - [Issue #426](#) Again meta data of library ports are screwed after insertion
 - [Issue #425](#) Connection via points not visible
 - [Issue #424](#) Wrong path for tooltip for state machines editor tabs
 - [Issue #431](#) Test for recently opened state machine fails
 - [Issue #430](#) Selection test fails

2.63.1 Patch releases 0.11.*

2.64 0.11.6

- Bug Fixes:
 - [Issue #428](#) fix recursion problem in execution log viewer
 - [Issue #427](#) Middle click on state machine tab label close wrong state machine
 - [Issue #419](#) wrong outcome data in execution history
- Improvements:
 - [Issue #411](#) Allow outputting data from preempted states
 - drag'n drop with focus can be enabled and disabled by using the gui config flag DRAG_N_DROP_WITH_FOCUS
 - graphical editor add way points around the state for self transitions as support for the user
 - refactor state machines editor tab click methods and small fixing
 - better on double click focus by gaphas editor and now also triggered by state machine tree

2.65 0.11.5

- Bug Fixes: - [Issue #421](#) RAFCON does not remember window size after closing -> final part

2.66 0.11.4

- New Features:
 - Move into viewport: Double click on elements in several widgets cause the element to moved into the viewport (not yet supported by all widgets)
 - Usage of selection modifiers (e.g. <Ctrl>, <Shift>) should now be more consistent
 - Ports in the graphical editor can now be selection
 - The port selection is synchronized between the graphical editor and the other widgets
 - Ports can be removed from within the graphical editor
- Improvements:
 - Refactoring of the selection
 - Unit tests for selection
 - [Issue #411](#) Allow outputting data from preempted states
 - [Issue #410](#) Refactor selection
 - [Issue #403](#) Incomes and outcomes cannot be differentiated visually
- Bug Fixes:
 - Memory leak fixes
 - [Issue #402](#) Connections end in nowhere
 - [Issue #417](#) ports of root state do not move with roots state
 - [Issue #421](#) RAFCON does not remeber window size after closing -> first part

2.67 0.11.3

- Improvements:
 - [Issue #405](#) Possibility to zoom in and out while drawing a connection
 - [Issue #404](#) Possibility to scroll left and right in graphical editor
 - [Issue #403](#) Incomes and outcomes cannot be differentiated visually
- Bug Fixes:
 - [Issue #412](#) global variables cannot be removed
 - [Issue #413](#) tree view controller error

2.68 0.11.2

- Improvements:
 - meta data scaling more robust and protect other elements from side effects of it
- Bug Fixes:
 - [Issue #393](#) \$HOME/.config/rafcon is not generated initially + tests
 - [Issue #406](#) Empty library root state without child states cause meta data resize problems with side effects in gaphas drawing

2.69 0.11.1

- New Features:
 - [Issue #384](#) add “Collapse all” button for library manager and enable the feature for the state machine tree, too
- Improvements:
 - port position default values
- Bug Fixes:
 - Fix issues when copying/converting logical or data ports with clipboard while cut/copy/paste
 - Fix library state port position scaling after adding
 - Fix gaphas viewer problems with undo/redo of complex actions like copy and paste or add/remove of ports
 - [Issue #10](#) Fully integrate modification history with gaphas

2.70 0.11.0

- New Features:
 - “Session restore” by default enabled
 - [Issue #364](#) “Open Recent” recently opened state state machines sub menu in menu bar under sub-menu Files
 - “Save as copy” in menu bar under sub-menu Files
 - “Show library content” supported for gaphas graphical viewer
 - The inner library states can be selected, copied and used to run the execution from or to this state, see [Issue #366](#) and [Issue #367](#), too
 - [Issue #255](#) The state machine tree shows inner library states, too, and can be used to explore all “leaf”-states
 - Storage format can be adapted by the user (e.g. names of states in paths and there length)
 - The library manager widget/tree supports modifications by right click (remove library, add/remove library roots)
 - Execution tool-bar supports buttons for run to- and run from-state (like right click menu, too)
- Improvements:

- Refactoring of “Save state as state machine/library”
 - Better default position meta data for states in graphical viewer
 - Proper resize of graphical meta data for complex actions and show library content
 - [Issue #369](#) Storage/Load module for state machines more flexible and robust
 - Storage module supports the user to store state machines without platform specific file system format conflicts
 - [Issue #365](#) substitute widget in now scrollable
 - The gtkmvc version 1.99.2 is fully supported ([Issue #388](#) corrected version in older releases)
- Bug Fixes:

[Issue #382](#) Currently active state machine not correct [Issue #362](#) Data flows between scoped variables [Issue #354](#) Meta data broken when adding state as template to state machine [Issue #353](#) Label not shown when adding state from library

2.70.1 Patch releases 0.10.*

2.71 0.10.3

- Bug Fixes:
 - File Chooser crashed if the same folder was added to the shortcut_folders twice

2.72 0.10.2

- Bug Fixes:
 - [Issue #385](#) If runtime config is newly created the last open path is empty and now state machine could be saved

2.73 0.10.1

- Bug Fixes:
 - make execution logs compatible with execution log viewer again

2.74 0.10.0

- Improvements:
 - complex actions(copy & paste, resize) are properly handled in gaphas and in the modification history
 - [Issue #342](#) drag and drop now drops the state at the mouse position
- Bug Fixes:
 - show library content for OpenGL works again
 - add as template works again

- [Issue #343](#) Text field does not follow cursor

2.74.1 Patch releases 0.9.*

2.75 0.9.8

- Improvements:
 - execution history can be logged and is configurable via the config.yaml
- Bug Fixes
 - [Issue #349](#) Save as library functionality erroneous
 - [Issue #314](#) Recursion limit reached when including top statemachine as replacement for missing state machine
 - [Issue #341](#) Reload only selected state machine
 - [Issue #339](#) Only save the statemachine.json
 - [Issue #338](#) Selecting a library state should show the data ports widget per default
 - [Issue #327](#) State machines are not properly selected
 - [Issue #337](#) Pressing the right arrow in the state machine editor opens a new state machine
 - [Issue #346](#) Barrier State cannot be deleted

2.77 0.9.6

- Bug fixes
 - fix step mode

2.78 0.9.5

- Bug fixes
 - runtime value flag of library states can be set again
 - add missing files of last release

2.79 0.9.4

- Bug Fixes
 - change VERSION file install rule to: ./VERSION => ./VERSION

2.80 0.9.3

- Bug Fixes
 - Fix missing VERSION file

2.81 0.9.2

- Improvements
 - Add rmpm env test
 - First version of setup.py
 - Version determination now in rafcon.__init__.py
 - Add another plugin hook, which is called each time a state machine finishes its execution
- Bug Fixes
 - Fix complex issues including the decider state
 - [Issue #322](#) Group/Ungroup is not working when performed on childs of a BarrierConcurrencyState
 - [Issue #326](#) RAFCON_INSTANCE_LOCK_FILE exception

2.82 0.9.1

- Bug Fix - fix bad storage format in combination with wrong jsonconversion version

2.83 0.9.0

- Improvements
 - Consistent storage format
 - Renamed modules: mvc to gui and core to statemachine
 - External editor
 - Substitute State
 - Open externally
 - Save selected state as library
 - Meta data convert methods with clear interface from Gaphas to OpenGL and OpenGL to Gaphas -> only one type of meta data hold

- Undocked side bars can be restored automatically after restart if *RESTORE_UNDOCKED_SIDEBARS* is set to True.
- Bug Fixes
 - Issue #299: State labels can be placed outside the state borders
 - Issue #298: Child states can be placed outside hierarchy states
 - Issue #45: Size of GUI cannot be changed
 - Issue #284: Core does not check the type of the default values
 - Issue #282: Input and output data port default_value check does not cover all cases
 - Issue #280: List of tuples saved as list of lists
 - Issue #265: jekyll documentation
 - Issue #277: insert_self_transition_meta_data is never called
 - Issue #268: Enter key can still be used in greyed out window
 - Issue #69: Performance measurements
 - Issue #271: The storage folders are not always clean after re-saving a state machine from old format to new
 - Issue #273: Cannot refresh state machines
 - Issue #264: pylint under osl not working
 - Issue #173: Splash screen for RAFCON GUI initialization and RAFCON icon
 - Issue #253: Ctrl+V for pasting in list views of state editor does not work
 - Issue #263: The scrollbar in the io widget has to follow the currently edited text
 - Issue #255: After refreshing, state machines should keep their tab order
 - Issue #185: test_backward_stepping_barrier_state not working
 - Issue #258: Maximum recursion depth reached
 - Issue #245: Support library data port type change
 - Issue #251: Handles are added when hovering over a transition handle
 - Issue #259: Do not hard code version in about dialog
 - Issue #260: Meta data is loaded several times

2.83.1 Patch releases 0.8.*

2.84 0.8.4

- Improvements: - allow loading of state machines created with RAFCON 0.9.*

2.85 0.8.3

- Bug Fixes: - fix copy paste of library states, consisting of containers - fix error output of not matching output data types

2.86 0.8.2

- Bug Fixes: - fix copy and paste for ports - fix backward compatibility test

2.87 0.8.1

- Features:
 - renaming of module paths: core instead of state machine; gui instead of mvc
 - writing wrong data types into the outputs of the “execute” function produces an error now
 - Use external source editor: A button next to the source editor allows to open your code in an external editor, which you can configure
 - Gaphas: When resizing states, grid lines are shown helping states to be aligned to each other (as when moving states)
- Improvements:
 - Gaphas: Change drawing order of state elements. Transitions are now drawn above states, Names of states are drawn beneath everything. This should ease the manipulation of transitions.
 - Gaphas: States are easier to resize, as the corresponding handle is easier to grab
 - states are now saved in folder that are named after: state.name + \$ + state.state_id
- API:
 - library paths can now be defined relative to the config file (this was possible before, but only if the path was prepended with “./”)
- Documentation:
 - started creation of “Developer’s Guide”
 - moved odt document about commit guidelines into rst file for “Developer’s Guide”
- Fixes:
 - Issue #5: Fix connection bug
 - Issue #120: Make state machines thread safe using RLocks
 - Issue #154: Multi-Selection problems
 - Issue #159: Transitions cannot be selected
 - Issue #179: Allow external source editor
 - Issue #202: RAFCON crash
 - Issue #221: issue when dragging data flows
 - Issue #222: Cannot remove transition of root state in TransitionController

- Issue #223: rafcon library config relative path undefined behaviour
- Issue #224: Switch to respective state when trying to open a state which is already open.
- Refactoring:
 - Widgets have TreeViews not have a common base class. This allowed to get rid of a lot of duplicate code and made some implementations more robust
 - the code behind connection creation and modification in the Gaphas editor has been completely rewritten and made more robust

2.88 0.8.0

- deactivated as not compatible with 0.7.13

2.88.1 Patch releases 0.7.*

2.89 0.7.13

- states are now saved in folder that are named after: state.name + \$ + state.state_id
- Hotfix: - fix unmovable windows for sled11 64bit

2.90 0.7.12

- Features:
 - Bidirectional graphical editor and states-editor selection with multi-selection support
 - Linkage overview widget redesign for optimized space usage and better interface
- Improvements:
 - Global variable manager and its type handling
 - Configuration GUI and its observation
 - State substitution: preserve default or runtime values of ports
 - Group/ungroup states
 - LibraryManager remembers missing ignored libraries
 - New config option LIBRARY_TREE_PATH_HUMAN_READABLE: Replaces underscores with spaces in Library tree
 - Update of transition and data flow widgets
- API:
 - ExecutionHistory is now observable
 - Configurations are now observable
 - allow to set `from_state_id` id `add_transition` method for start transitions
- Fixes

- Issue #177: Data flow hiding not working
 - Issue #183: Rafcon freeze after global variable delete
 - Issue #53: Configurations GUI
 - Issue #181: State type change not working
 - Several further fixes
- Refactorings, optimizations, clean ups

2.91 0.7.11

- Features:
 - Global variables can now be typed, see [Feature #81](#)
 - GUI for modifying the configurations
 - Config files can be im- and exported
 - Graphical editor can be shown in fullscreen mode (default with F11), see [Feature #36](#)
 - I18n: RAFCON can be translated into other languages, rudimentary German translation is available
 - RAFCON core can be started with several state machines
- Improvements:
 - Fix backward compatibility for old `statemachine.yaml` files
 - Issue #136: Undocked sidebars no longer have an entry in the task bar and are shown on top with the main window
 - Added tooltips
 - When starting RAFCON from the console, not only the path to, but also the file name of a config file can be specified. This allows several config files to be stored in one folder
 - Use correct last path in file/folder dialogs
 - Show root folder of libraries in the shortcut folder list of file/folder dialogs
 - new actions in menu bar, menu bar shows shortcuts
 - Source and description editor remember cursor positions
- API:
 - State machines and their models can be hashed
- Fixes
 - Issue #161: When refreshing a running state machine, the refreshed one is still running
 - Issue #168: Undocked sidebars cause issues with `is_focus()`
 - Issue #169: Wrong dirty flag handling
 - Issue #182: Test start script waits infinitely
 - Several further fixes
- Refactorings, optimizations, clean ups

2.92 0.7.10

- Features
 - State substitution
 - Right click menu differentiate between states and library states
- Improvements
 - Graphical editor Gaphas:
 - way faster
 - more stable
 - connections are drawn behind states
 - small elements are hidden
 - BuildBot also runs tests on 32bit SLED slave
 - Core documentation
- Issues fixed
 - Issue #143
 - Issue #139
 - Issue #146
 - Issue #145
 - Issue #122
 - Issue #149
 - Issue #119
 - Issue #151
 - Issue #155
 - Issue #17
- Lots of further fixes and improvements

2.93 0.7.9

- Features:
 - Grouping and ungrouping of states
 - Initial version of possibility to save arbitrary states as libraries and to substitute one state with another one
 - Right click menu for graphical editor
 - add flags to `mvc.start.py`
- Bug fixes
 - Issue #132
 - Issue #40
 - Issue #65

- Issue #131
- Issue #105
- Kill RAFCON with Ctrl+C
- Resizing of states in Gaphas
- Correctly distinguish string and unicode data port types when using library states (should fix issues with ROS)
- Stepping starts a state machine if not started
- Improvements
 - Gaphas works more reliable, especially concerning copy'n'paste and selection
 - History
- Some changes in destruction hooks
- Refactorings
 - Many for Gaphas components, e.g. the border size of a state depends on the state size now
 - Obsolete models are deleted (=> less memory consumption)
 - Remove state_helper.py
- New network tests
- Add missing GUI drafts of Jürgen

2.94 0.7.8

- Add tests
- ExecutionEngine: Notify condition on all events except pause

2.95 0.7.7

- Add three new hooks
 - `main_window_setup`: Passes reference to the main window controller and is called after the view has been registered
 - `pre_main_window_destruction`: Passes reference to the main window controller and is called right before the main window is destroyed
 - `post_main_window_destruction`: is called after the GTK main loop has been terminated

2.96 0.7.6

- remove obsolete files
- properly destruct states on their deletion (+ test to check unctionality)
- jump to state on double-click in ExecutionHistory
- fixes in display of ExecutionHistory
- fix not shown description of LibraryStates
- fix crash on middle-click on state machine tab
- Fix copy & paste of ExecutionStates
- improve tests
- improve documentation (add missing elements)
- Show ‘+’ for adding state machines
- example on abortion handling
- Add config option to hide data flow name
- Fix [Issue #129](#)
- get rid of all plugin dependencies
- no more need to change into the mvc-directory when working with the GUI
- refactoring (especially in start.py)
- more fixes

2.97 0.7.5

- Improve Execution-History visualization with proper hierarchical tree view and improved data and logical outcome description (on right-click)
- Improve auto-backup and add lock files to offer formal procedure to recover state machine from temporary storage (see [Auto Recovery](#))
- Improve Description editor by undo/redo feature similar to the SourceEditor
- Improve versions of “monitoring” and “execution hooks” plugins
- Improve graphical editor schemes (OpenGL and Gaphas) and Gaphas able to undo/redo state meta data changes
- Introduce optional profiler to check for computation leaks in state machine while execution
- Bug fixes

2.98 0.7.4

- Improve performance of GUI while executing state machine with high frequent state changes
- Fix [Issue #121](#) Properly copy nested ExecutionStates

2.99 0.7.3

- States are notified about pause and resume (See [FAQ](#) about *preemption* and *pause*)
- Load libraries specified in RAFCON_LIBRARY_PATH (See [this tutorial](#))
- improve stability
- refactorings
- bug fixes

2.100 0.7.2

- improved auto-backup to tmp-folder
- fix missing logger messages while loading configuration files
- introduced templates to build plugins
- re-organized examples to one folder -> share/examples, with examples for API, libraries, plugins and tutorials
- introduce short-cut for applying ExecutionState-Scripts
- smaller bug fixes

2.101 0.7.1

- Allow multiple data flows to same input data ports (in order to remain backward compatibility)

2.102 0.7.0

This is a big minor release including many changes. State machines stored with version 0.6.* are compatible with this version, but not state machines from older releases. Those have to be opened with 0.6.* and then saved again. The following list is probably not complete:

- Support for openSUSE Leap
- Support for plugins
- Major design overhaul: agrees with drafts from design and looks consistent on all platforms
- Drag and Drop of states
 - Libraries from the library tree
 - Any type of state from the buttons below the graphical state editor
 - The drop position determines the location and the parent of the new state

- All sidebars can now be undocked and moved to another screen
- Auto store state machine in background and recover after crash
- Improved history with branches
- New feature: run until state
- Extended stepping mode: step into, over and out
- Redesign remote execution of state machines: Native GUI can be used to execute state machine running on different host
- Drop support of YAML state machine files
- Rename state machine files
- Extend documentation
- RMC-BuildBot support
- Many bug fixes
- A lot of refactorings, code optimizations, etc.

2.102.1 Patch releases 0.6.*

2.103 0.6.0

- Prepare code and folder structure to allow theming (currently only dark theme available)
- Refactor GUI configuration and color handling
- Fix network_connection initialization
- Use python2.7 by default when using RAFCON with RMPM
- Gaphas graphical editor:
 - change cursor when hovering different parts of the state machine
 - add hover effect for ports
 - no more traces of states/labels when moving/resizing states/ports
 - resize handles are scaled depending on zoom level and state hierarchy
 - do not show handles on lines that cannot be moved
 - improve behavior of line splitting
 - refactorings
 - minor bug fixes
- Fix many code issues (line spacing, comments, unused imports, line length, ...)
- fix bug in global variable manager, causing casual exception when two threads access the same variable

2.103.1 Patch releases 0.5.*

2.104 0.5.5

fix start from selected state (the start-from-selected-state functionality modifies the start state of a hierarchy state on the initial execution of the statemachine; the start state was accidentally modified for each execution of the hierarchy state during one run leading to wrong execution of hierarchy states that were executed more often during the execution of a statemachine)

2.105 0.5.4

hotfix for mvc start.py launching with network support enabled

2.106 0.5.3

hotfix for rafcon server

2.107 0.5.1 + 0.5.2

feature: command line parameter to start state machine at an arbitrary state

2.108 0.5.0

- State-machines can be stored in JSON files instead of YAML files
 - Set USE_JSON parameter in config to True
 - Loads state-machines approximately five times faster
- Removed some code ensuring backwards compatibility of old state-machines
 - If you are having trouble loading older state-machines, open them with the last version of the 0.4.* branch
 - Save them and try again with the 0.5.* branch

2.108.1 Patch releases 0.4.*

2.109 0.4.6

- Add start scripts in bin folder
- When using RAFCON with RMPM, you can run RAFCON just with the commands `rafcon_start` or `rafcon_start_gui`
- Bug fixes for state type changes

2.110 0.4.5

- Feature: Add late load for libraries
- State type changes work now with Gaphas graphical editor
- Minor code refactorings

2.111 0.4.4

- Fix bug: changing the execution state of a statemachine does mark a statemachine as modified

2.112 0.4.3

- Fix bug: data port id generation
- Fix bug: runtime value handling

2.113 0.4.2

- Feature: runtime values

2.114 0.4.1

- Fix bug: resize of libraries when loading state machine
- Fix bug: error when adding data port to empty root state

2.115 0.4.0

- Show content of library states
- Keep library tree status when refreshing library
- Allow to easily navigate in table view of the GUI using the tab key
- Refactor logger (new handlers) and logger view
- Many refactorings for Gaphas graphical editor
- Introduce caching for Gaphas graphical editor => big speed up
- Require port names to be unique
- Highlight tab of running state machine
- Default values of library states can be set to be overwritten
- Improve dialogs
- make meta data observable

- many bug fixes
- clean code
- ...

2.115.1 Patch releases 0.3.*

2.116 0.3.7

- rafcon no-gui start script also supports BarrierConcurrency and PreemptiveConcurrencyStates

2.117 0.3.6

- bugfix if no runtime_config existing

2.118 0.3.5

- rafcon_server can be launched from command line
- network config can be passed as an argument on startup

2.119 0.3.4

- first version of rafcon server released

2.120 0.3.3

- state machines can be launched without GUI from the command line

2.121 0.3.2

- Extend and clean documentation (especially about MVC) and add it to the release
- Waypoints are moved with transition/data flows (OpenGL editor)
- data type of ports of libraries are updated in state machines when being changed in the library
- bug fix: error when moving waypoint
- bug fix: add new state, when no state is selected

2.122 0.3.1

- Support loading of old meta data
- bug fix: errors when removing connected outcome
- bug fix: network config not loaded
- code refactoring: remove old controllers, consistent naming of the rest

2.123 0.3.0

- RAFCON server to generate html/css/js files for remote viewer (inside browser)
- optimize workflow:
 - root state of new state machines is automatically selected
 - new states can directly be added with shortcuts, without using the mouse beforehand
 - A adds hierarchy state (A for execution states)
- support loading of state machines generated with the old editor in the new editor
- bug fixes for graphical editor using gaphas (especially concerning the state name)
- bug fixes for states editor

2.123.1 Patch releases 0.2.*

2.124 0.2.5

- update LN include script (use pipe_include and RMPM)
- allow configuration of shortcuts
- distinguish between empty string and None for ports of type str
- bug fixes in GUI (start state)

2.125 0.2.4

- introduce env variables RAFCON_PATH and RAFCON_LIB_PATH
- automatically set by RMPM

2.126 0.2.3

- use of separate temp paths for different users

2.127 0.2.2

- Allow RAFCON to be started from arbitrary paths

2.128 0.2.1

- minor code refactoring
- RMPM release test

2.129 0.2.0

- First release version
- Tool was renamed to RAFCON

**CHAPTER
THREE**

LIST OF FEATURES

In the following, all features of RAFCON are listed and categorized into either *Core*, *GUI*, *Consistency Check* or *Misc*. All listed features have their own, dedicated unit tests.

3.1 Core Features

Features	Notes
State Machines	Formal definition
Hierarchies	For easy behavior modularization
Concurrents	Barrier and preemptive paradigm
Library States (Libraries)	For code reusability
Data Flows + Data Ports	For explicit data modeling
Default Values for Data Ports	
Scoped Variable Concept	Private, scoped and global data
Every State Machine can be used as Library	Without any overhead
Execution Capabilities:	IDE like debugging mechanism
<ul style="list-style-type: none"> • Start, Stop, Pause 	
<ul style="list-style-type: none"> • Step Mode: Into, Over, Out 	
<ul style="list-style-type: none"> • Backward Step 	Optionally definable in each state
<ul style="list-style-type: none"> • Execution History 	
<ul style="list-style-type: none"> • Execution History Logging 	
<ul style="list-style-type: none"> • Explicit Preemption Handling 	
<ul style="list-style-type: none"> • Error Propagation 	
<ul style="list-style-type: none"> • Execution Signals and Preemptive Waits 	
Remote Monitoring	Via Monitoring-Plugin
Dynamic Modifications Possible	
API for Programmatic State Machine Generation	
Customizable Logging Features	

3.2 GUI Features

Features	Notes
State Machine Visualizer	Using Gaphas
State Machine Editor	
<ul style="list-style-type: none"> • Creation of State Machines Elements 	

continues on next page

Table 1 – continued from previous page

Features	Notes
• Navigation inside State Machine incl. Zoom	
MVC Concept	
State Templates	For fast prototyping
Separate Visualization of Data Flows	
Rearrangeable Ports	
Waypoints for Connections	
Focusable States	
Executing States are Highlighted	
Embedded Source Code Editor	
Code Syntax Check	
Library Tree	
State Machine Tree	Enables easy state selection
Modification History	
Online Execution History	
State Description	Name, ID, type, link to library
Logging Console View	
Design by Professional Interface Designers	In GTK3, css will be used
Resizeable, Detachable and Foldable Widgets	
Restoring of User Specific GUI Layout	
Library Relocation Support Mechanism	
Dialog Boxes for User Communication	
Menu for Recently Opened State Machines	
Properties Window	
Session Restore	
Shortcut Manager	
Library Content Introspection	i.e. show-content-flag per library state
Editor Utility Functions:	
• Copy + Paste	
• Grouping and Ungrouping of States	
• Drag and Drop of States	
• State Type Change	
• Intelligent Auto State Connector	e.g. connect income/outcome with closest sibling

3.3 Consistency Checks

Features	Notes
Data Type Check	
Data Flow Validity	
Hierarchical Consistency	E.g. no connection to grandparents
Data Integrity Check During Runtime	
Errors in Python Scripts are Caught	And forwarded to the next states as datum
Library Interface Consistency	

3.4 Misc Feat

Features	Notes
State Machine Versioning	Via git
Human Readable File Format	json and yaml
Programmable in Python	Python 3
Middleware Independent	Tested with: ros, links and nodes, sensornet, and DDS
Core and GUI Separated	Core can run on micro-controller with slim Python setup
Extensive Documentation	Via Restructured Text (rst) and Sphinx
Plugin Concept	Several plugins available
Backward Compatibility	Breaking changes are clearly outlined
No Memory Leaks	See test_destruct.py in tests folder
Continuous Integration	Buildbot / Jenkins
Usable in Different Robotic Domains	Used in: Space, Industry, Service
Scalability: Tested with >4000 states	
Video Tutorials	Youtube (only one available, more to come)

**CHAPTER
FOUR**

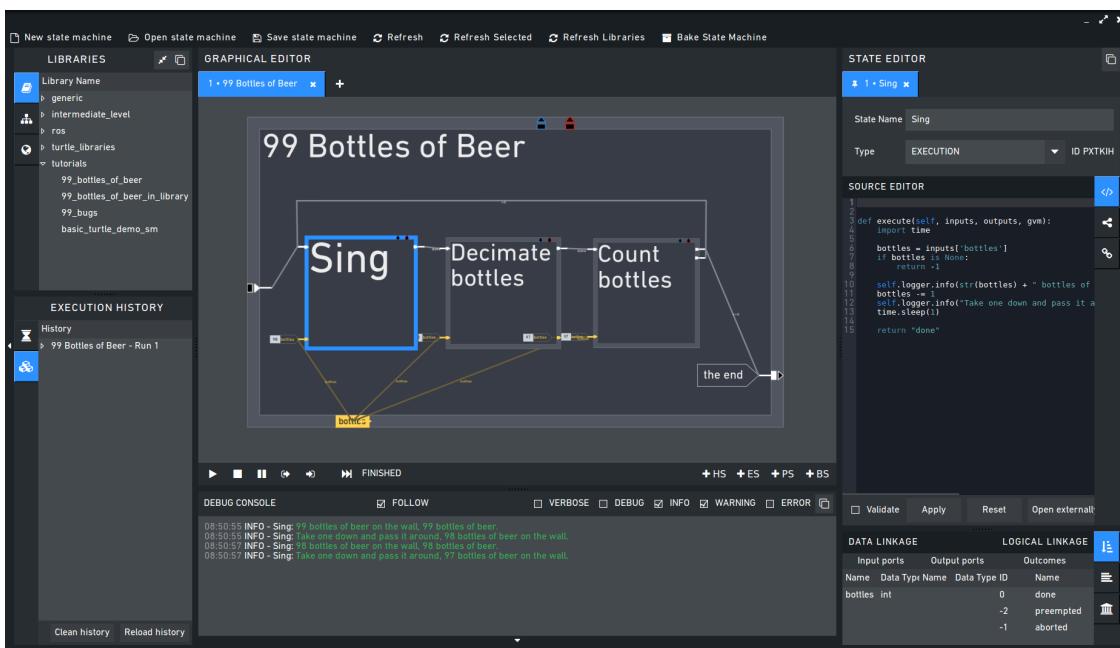
BREAKING CHANGES

The following gives a list of all RAFCON minor versions (starting with 0.12.x) and their ability to load state machines of other minor versions.

- 0.12.x: can load all state machines from at least version 0.9.x and up

TUTORIALS

5.1 99 Bottles of Beer - Or: How to create a simple state machine containing a loop



In this tutorial we create a simple state-machine printing the lyrics of “99 Bottles of Beer”, a slightly advanced version of a Hello World program. The purpose is to give a first impression of how to create programs with *RAFCON*, how to create loops and illustrate different ways of achieving things in *RAFCON*.

1. Start with a new state machine (first button in the tool bar or File => New in the menu bar). A new tab opens with the root container state named “new_root_state”.
2. Select the root state by clicking on it in the graphical editor (big center widget) or by opening the “State Tree” widget (left hand side) and clicking on the state here.
3. Now we change the name of the root state to “99 Bottles of Beer” by entering that text in the input field next to “State Name” in the State Editor on the right bar. Here you can also see all other properties of our container state.
4. Now we are going to create three execution states named “Sing”, “Decimate bottles” and “Count bottles”. Do so by selecting the container state first and then either use Alt+E or Edit > Add state in the menu bar. Rename each new state as described in the previous step.

5. You can use drag and drop for the three child states to place them next to each other. Using the lower center corner of all states, you can resize them. When holding Ctrl during the resize, the state keeps its width to height ratio. When holding Ctrl during the resize (of container states), all content is resized with the state.
6. Next, we are going to add the logical flow, that is the outcomes and transitions. “Sing” and “Decimate bottles” each need one outcome, which should exist by default. For “Count bottles”, we need two outcomes, the first (ID 0) is named “==0”, the second (ID 1) is named “>0”. Click on the state to open it in the State Editor and open the “Logical Linkage” widget on the bottom. In the upper half of this widget, you can edit the state’s outcomes. Create an additional outcome by clicking on “Add” or using Ctrl+A if the outcomes list has the focus. Then name the new outcomes (by clicking on the name in the name column). Also the container state needs an outcome, name it “the end”.
7. To finish the logical flow, we set up all transitions and start states. “Sing” is the start state of our only container (the root). Thus click on it and check “is start state” in the top of the State Editor. This creates a transition from the entry point of the root to the entry point of “Sing”. The easiest way to create the remaining transitions is to first left click on an outcome and, while holding the left mouse pressed, dragging the mouse over the income of the target state and release the mouse button there. This creates a transition starting from the outcome and going to the entry point of the state. The starting transition can be created in the same manner. Also waypoints for transitions can be created. In the default GUI (gaphas) waypoints can only be created after the transition has been created. Select any transition then click on the grey square and move it. Two new square emerge that can be modified accordingly. In the OpenGL GUI, you can set transition waypoints by clicking on the desired position within the container state while creating the transition. Later, you can add/remove waypoints by clicking on the desired position of the transition. By this, you can e. g. draw transitions around other states. Another possibility to create transitions is the State Editor again. In the lower half of the Logical Linkage widget, new transitions can be created by a click on “Add” (or using Ctrl+A if the focus is on the transition list). The widget tries to guess on which transition to create, but you can edit the origin and target with the dropdown list in the appropriate column. We need four transitions. One from “Sing” to “Decimate bottles”, then one from “Decimate bottles” to “Count bottles”, one from “Count bottles” outcome “>0” back to “Sing” and finally one from “Count bottles” outcome “==0” to “the end” of the container.
8. In this step we are going to create the data ports and scoped variables. For printing the verse, our “Sing” state needs the current number of bottles as input. Therefore we create an input data port. Select the “Sing” state and open the “Data Linkage” widget in the State Editor. The ports are handled in the upper half of this widget. First select the “Input Ports” tab and then create an input by clicking on the “New” button. Set the name to “bottles” and the type to “int”. We do not need a default value here (the default is automatically “None”). “Decimate bottles” needs an input and an output port to read in the current number of bottles and to return the new number of bottles. Thus, create the two ports, both named “bottles” and both of type “int” (again no default values). “Count bottles” also needs only an input port of type “int” and name “bottles”.
9. In order to hold data between the loop iterations, we need a scoped variable. This variable is defined in the container state. It is created analogous to inputs and outputs, just in the “Scoped Variables” tab. Name and type are again “bottles” and “int”. Here we set the default value, which is also the initial value, to 99. Scoped variables can be moved in the graphical editor just like states with drag and drop. To move the scoped variable the corresponding state has to be selected and Ctrl pressed.
10. The data flows are now created similar to the transitions. Either in the graphical editor by clicking on the origin port (from where to read) and then clicking on the target port (to where to write) or, alternatively the bottom half of the Data Linkage widget can be used. Create the following data flows:
 1. From the scoped variable to the input of “Sing”: Here we are reading the current number of bottles
 2. From the scoped variable to the input of “Decimate bottles”: Here we are reading the current number of bottles
 3. From the output of “Decimate bottles” to the scoped variable: Here we are writing the decimated number of bottles back
 4. From the output of “Decimate bottles” to the input of “Count bottles”: Here we directly pass the decimated

number to “Count bottles”. This could also have the scoped variable as origin.

11. Finally, we have to add some source code to the three child states. The code executed for each state is shown in the source code widget of the State Editor. The method description is automatically created. You just have to insert your code after the line `def execute(...)`. Copy the code following at the end of the tutorial into the states. Important: You have to click on “Apply” to apply changes. The `sleep` statements in the code only serve illustrative purposes, better visualizing the flow of active states. You see that you can read from ports using the inputs dictionary (`bottles = inputs['bottles']`) and write to ports using the outputs dictionary (`outputs['bottles'] = inputs['bottles'] - 1`). You can also rename these dictionaries if you prefer a different (shorter) name: `def execute(self, in, out, gvm):`. The logger is a member of each state and can be used to write to the logger window. You can use different logger levels (debug, info, warn, error, verbose). Moreover, custom logger levels can be created in the logger config file (`logging.conf` in source/rafcon).
12. You can now test the state machine. Click on Execution > Start and see what happens. The current state should be highlighted and the verses printed in the logger widget. You can stop the execution with Execution > Stop. Alternatively, use the buttons shown in the menu of the Graphical Editor.

```
# State "Sing"
def execute(self, inputs, outputs, gvm):
    import time

    bottles = inputs['bottles']
    if bottles is None:
        return -1

    self.logger.info(str(bottles) + " bottles of beer on the wall, " + str(bottles) + " "
                     + bottles + ".")
    bottles -= 1
    self.logger.info("Take one down and pass it around, " + str(bottles) + " bottles of "
                     + bottles + ".")
    time.sleep(1)

    return 0
```

```
# State "Decimate bottles"
def execute(self, inputs, outputs, gvm):
    import time

    outputs['bottles'] = inputs['bottles'] - 1
    time.sleep(0.2)

    return 0
```

```
# State "Count bottles"
def execute(self, inputs, outputs, gvm):
    import time

    time.sleep(0.2)

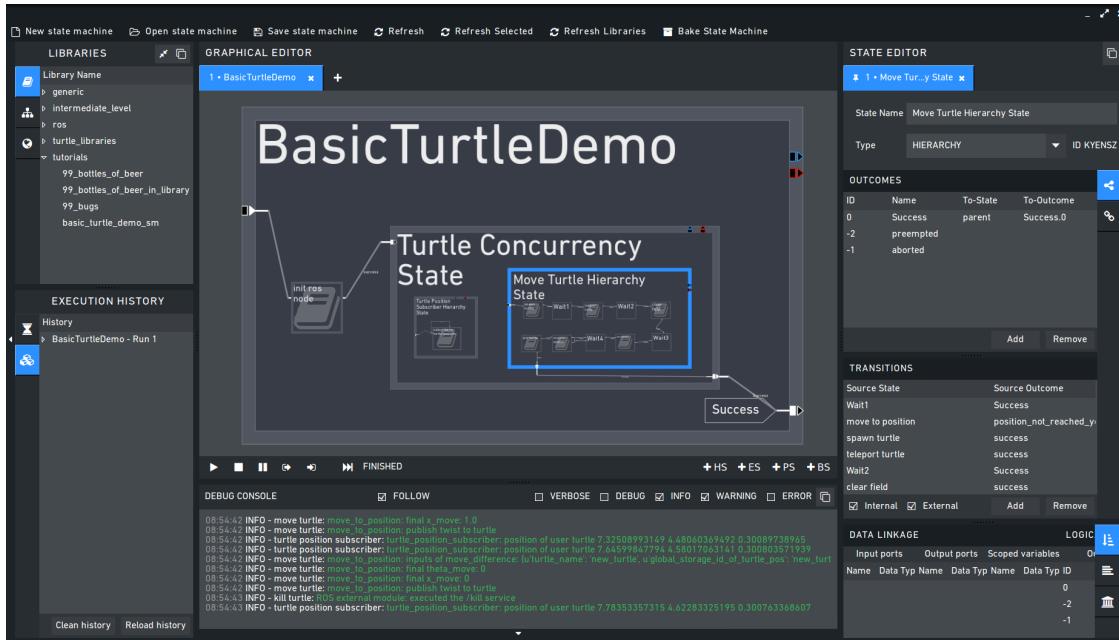
    if inputs['bottles'] > 0:
        return 1
    return 0
```

5.2 Starting the basic turtle demo state machine using ROS

The basic turtle demo is a demo to demonstrate the use of libraries and to show the easy integration of ROS into the RAFCON. To start the turtle demo just open the basic_turtle_state_machine in the tutorials library folder and click on start. The following code blocks include code lines to generate the correct environment for our institute PCs; in an e.g. Ubuntu setup, where the environment is statically specified in the `~/.bashrc` these environment generating commands can be omitted:

```
rmpm_do env ros.indigo.desktop > /tmp/desktop.env
source /tmp/desktop.env
rmpm_do env rafcon > /tmp/rafcon.env
source /tmp/rafcon.env
cd ${RAFCON_GIT_HUB_REPO_OR_RMPM_PATH}/share/examples/api/generate_state_machine
python basic_turtle_state_machine.py
```

A screenshot of how the state machine looks like is shown here.



Next start a roscore in another console:

```
rmpm_do env ros.indigo.desktop > /tmp/desktop.env
source /tmp/desktop.env
roscore
```

And the turtlesim node in yet another console:

```
rmpm_do env ros.indigo.desktop > /tmp/desktop.env
source /tmp/desktop.env
rosrun turtlesim turtlesim_node
```

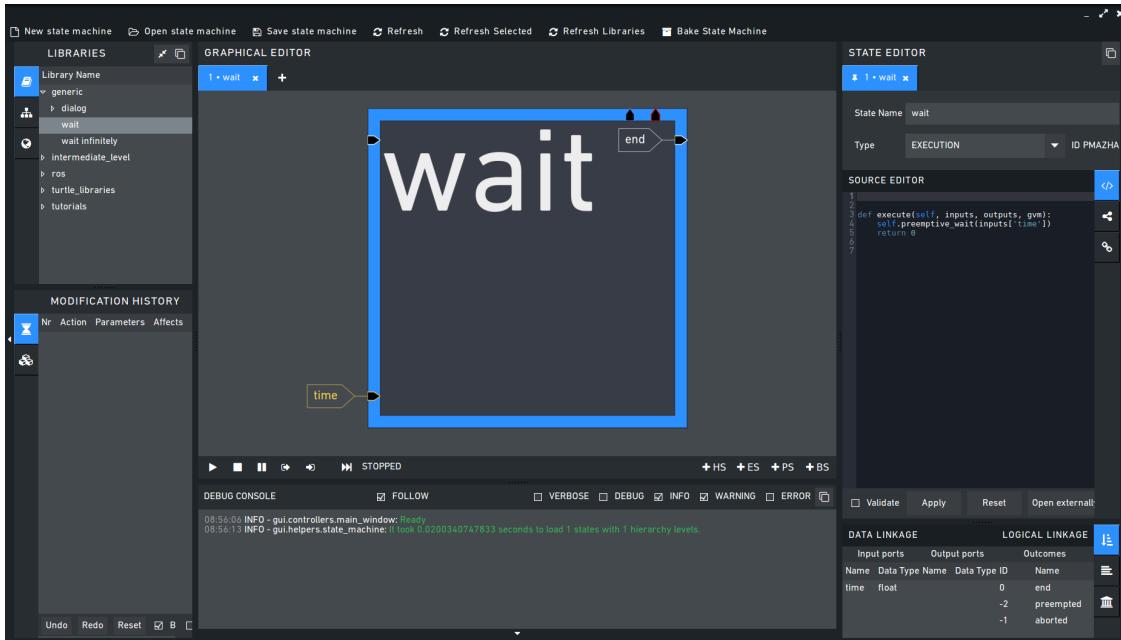
After that start the state machine. The state machine will then start some basic services of the turtlesim in a sequence.

5.3 How to create and re-use a library state machine

State machines can easily be reused in form of library states. All you have to do for this is telling RAFCON the path to your state machine and give this path a name.

5.3.1 Option 1

This can be done in the *Core Configuration*.



Let's add a new library path to our config file, which by default looks like this:

```
TYPE: SM_CONFIG
LIBRARY_PATHS:
    generic: ${RAFCON_LIB_PATH}/generic
USE_JSON: true
```

We edit the LIBRARY_PATH to take into account the library with name “lib_tutorial” located at `~/Desktop/rafcon_tutorial_library`:

```
TYPE: SM_CONFIG
LIBRARY_PATHS:
    generic: ${RAFCON_LIB_PATH}/generic
    lib_tutorial: ~/Desktop/rafcon_tutorial_library
USE_JSON: true
```

RAFCON assumes the path to be existing, so make sure it is. Otherwise RAFCON will print a warning message.

On the left side in the Library Tree, you can now see the new entry `lib_tutorial`, which is currently empty.

Next, we generate two state machines, one is waiting and another is printing a message to the logger console (info level). Generate two state machines by clicking the button “New state machine” and turn the root_state to a ExecutionState (by using StateEditorWidget on the center site and select “Execution” as type instead of “Hierarchy”) and insert the following scripts.

First:

```
import time

def execute(self, inputs, outputs, gvm):
    time = inputs['time']
    if self.preemptive_wait(time):
        return 'preempted'
    return 0 # same as return "success"
```

Second:

```
def execute(self, inputs, outputs, gvm):
    message_to_print = inputs['info_message']
    self.logger.info(message_to_print)
    return 0
```

Don't forget to create the input data ports used in the scripts ('time' as float and 'info_message' as string) and run them finally to test their functionality.

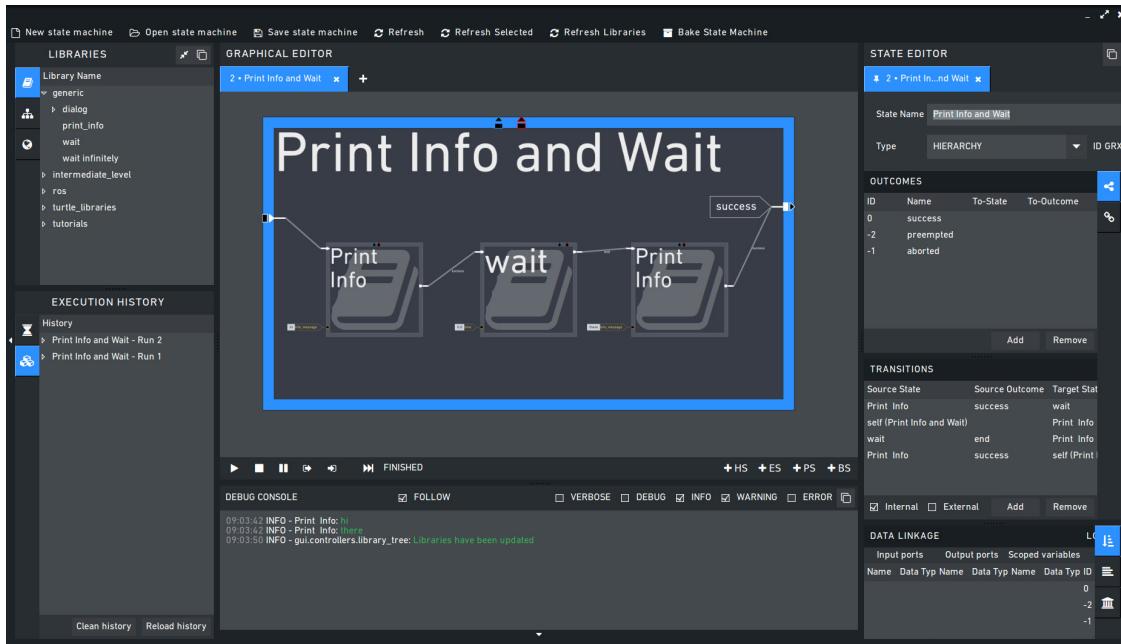


Fig. 1: Screenshot of the finished library tutorial

Give the state machines useful names like "Wait" for the first and "Print Info" for the second state machine.

Store both state machines (by pressing button "Save state machine" or Ctrl+s) to sub-folders of `~/Desktop/rafconTutorial/library` by entering a the library folder and assigning a name in the dialog window. The name is used to generate the new library state machine path.

Now press the button "Refresh Libraries". The new libraries will be now available in the library tree. They can be used to create more complex state machines.

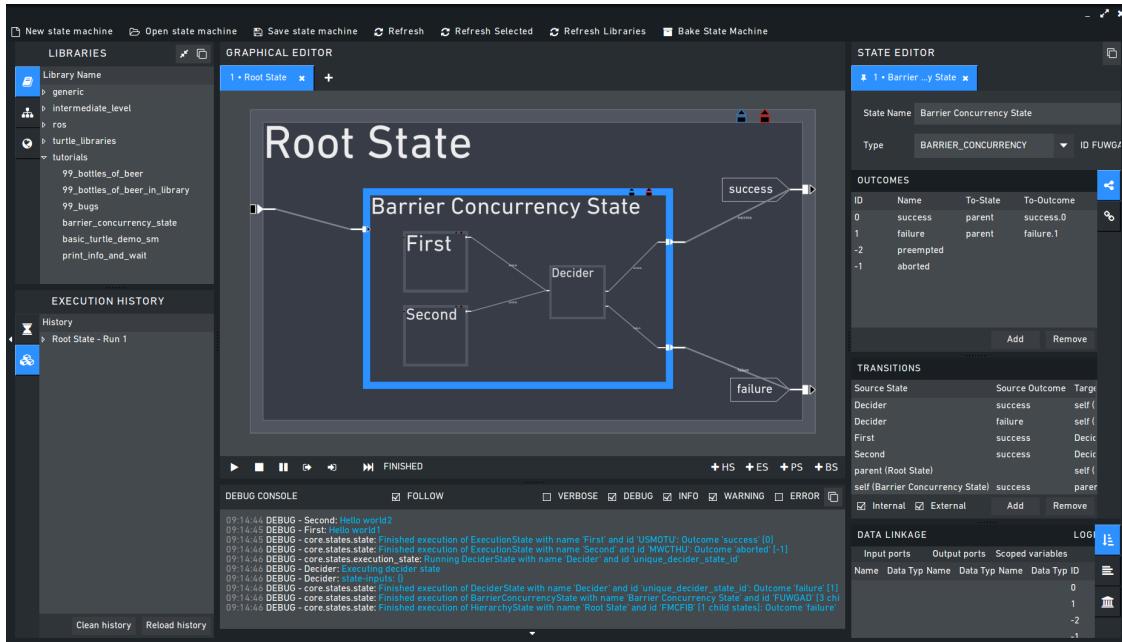
Using Drag&Drop, the created library state machines can be re-combined as in the "Screenshot of the finished library tutorial" and the input port values can be modified to generate similar console info prints while running the state machine.

5.3.2 Option 2

Instead of specifying the path of the library in the config file, there is an alternative solution. You can also set the environment variable `RAFCON_LIBRARY_PATH` being a colon-separated list of paths to state machines, e.g. `~/path/to/your/rafcon_tutorial_library1:~/path/to/your/rafcon_tutorial_library2`. These libraries will also be loaded. The name of the mounted library root keys is equivalent to name of the last folder of each path. In our case this would be `rafcon_tutorial_library1` and `rafcon_tutorial_library2`. This approach is especially useful if you use RAFCON in combination with a package management system such as conan (<https://conan.io/>) or a local pip server.

5.4 How to use concurrency barrier states

In the following a short example on how to create a barrier concurrency state is explained.



At first create the state and transition structure shown in the above image. The State called “Barrier Concurrency” is a barrier concurrency state. The state called decider is the state that is automatically created when a new barrier concurrency state is added. The decider state gets the information of all concurrent child states about the chosen outcome, the output data and even every eventually occurred error. Of course data flows can also arbitrarily be connected to the decider state from each concurrent child state. With this information it can decide via which outcome the barrier concurrency state is left.

To get some output paste the following source lines into the appropriate states:

First:

```
import time

def execute(self, inputs, outputs, gvm):
    time.sleep(1.0)
    self.logger.debug("Hello world1")
    return 0
```

Second:

```
import time

def execute(self, inputs, outputs, gvm):
    self.logger.debug("Hello world2")
    time.sleep(2.0)
    number = 1/0 # create an error here that can be handled in the decider state
    return 0
```

Decider:

```
from exceptions import ZeroDivisionError

def execute(self, inputs, outputs, gvm):
    self.logger.debug("Executing decider state")
    self.logger.debug("state-inputs: {}".format(str(inputs)))
    # to make decisions based on the outcome of the concurrent child states use:
    # "self.get_outcome_for_state_name(<name_of_state>)" for accessing the outcome by
    # specifying the name (not necessarily unique, first match is used) of the state
    # or "self.get_outcome_for_state_id(<id_of_state>)" for accessing the outcome by
    # specifying the id (unique) of the state
    # example:
    # if self.get_outcome_for_state_name("Second").name == "success":
    #     return 0
    # here the error of the state "Second" is used to make a decision
    if isinstance(self.get_errors_for_state_name("Second"), ZeroDivisionError):
        return 1
    else:
        return 0
```

5.5 Starting a minimal RAFCON core (RAFCON API)

This tutorial will show how to set up a minimal RAFCON core and use RAFCON API functionality to run state machines. The following script gives an overview of a basic setup. By saving it in a .py file, it can simply be executed afterwards. Note that the path to the config.yaml has to be set correctly. By default, it should be under the .config directory, as specified below. Similarly, the path_to_state_machine must point to an already existing state machine. In the example below it will execute the “99 Bottles of Beer”.

```
#!/usr/bin/env python3

import time

import rafcon.core.singleton as rafcon_singletons
import rafcon.core.start as rafcon_start

from rafcon.core.config import global_config as rafcon_global_config
from rafcon.core.execution.execution_status import StateMachineExecutionStatus as_
    ExecutionStatus
from rafcon.core.state_machine import StateMachine
from rafcon.core.storage import storage as rafcon_storage

def main():
```

(continues on next page)

(continued from previous page)

```

print("Initialize RAFCON ... ")
rafcon_start.pre_setup_plugins()
rafcon_start.setup_environment()
rafcon_start.setup_configuration("/home/user/.config/rafcon/config.yaml")
rafcon_global_config.set_config_value("EXECUTION_LOG_ENABLE", True)

print("Set and load state machine ... ")
path_to_state_machine = '/home/user/rafcon/source/rafcon/share/rafcon/examples/
˓→tutorials/99_bottles_of_beer/statemachine.json'
print(f"Start loading the statemachine: {path_to_state_machine}")
start_time = time.time()
state_machine = rafcon_storage.load_state_machine_from_path(path_to_state_machine)
stop_time = time.time()
diff = stop_time - start_time
print(f"Duration of loading: {diff}")

print("Set global variables ... ")
gvm = rafcon_singletons.global_variable_manager
gvm.set_variable(key='test_var', value=42)

print("Activate and start state machines")
execution_engine = rafcon_singletons.state_machine_execution_engine
state_machine_manager = rafcon_singletons.state_machine_manager
sm_id = state_machine_manager.add_state_machine(state_machine)
state_machine_manager.active_state_machine_id = sm_id

execution_engine.start()

if __name__ == "__main__":
    main()

```

More information on how to use the API can further on be found under *RAFCON API: The rafcon package*.

5.6 Using the monitoring plugin

The tutorial is only for internal use inside the institute.

This tutorial will show how to use the monitoring plugin i.e. how to monitor one system from another one if both are using RAFCON as their flow control solution. First, we need to setup our environment:

```
rmpm_do env rafcon_monitoring_plugin > /tmp/rafcon_monitoring_plugin.env
source /tmp/rafcon_monitoring_plugin.env
```

By running RAFCON after sourcing the environment, the `network_config.yaml` is automatically generated in our home folder: `~/.config/rafcon/` if it does not already exist. This file contains all settings for the communication. More details can be found at the [Configuration](#). The path of the `network_config.yaml` can be changed by running the `start.py` script with argument “-nc”, which will be necessary when we want to connect server and client running on a single system like in this tutorial. Therefore we create the subdirectories `/client` and `/server` within the `~/.config/rafcon/` path and copy/paste the `network_config.yaml` into both. Since the file is created for servers by default, we just have to edit the one in the `/client` directory, where we replace the `<SERVER: true>` column by `<SERVER: false>`.

Now we can launch the server:

```
rafcon -nc ~/.config/rafcon/server
```

and the client:

```
rafcon -nc ~/.config/rafcon/client
```

If everything went fine, we should see below output in the debug console of the client:

```
11:23:40 INFO - monitoring.client: Connect to server ('127.0.0.1', 9999)!  
11:23:40 INFO - monitoring.client: self.connector <monitoring.client.MonitoringClient on  
→59055>  
11:23:40 INFO - monitoring.client: sending protocol 34ce956f:72f0dc:2:4:Registering  
11:23:40 INFO - monitoring.client: Connected to server!
```

After the connection was established, we open the same state machine on server and client. Now we are able to remote control the server by the client. To connect two systems distributed across a network, the <SERVER_IP:> has to be adjusted within the `network_config.yaml` files.

5.7 How to use dialog states from the generic library

Sometimes it can be useful to await user confirmation before jumping into a state or request a text input from the user. That is why RAFCON contains several dialog states in its ‘generic’ library. This tutorial goes through several of them and explains their characteristics.

5.7.1 MessageDialog

This dialog prompts the user with a text which is defined by the string type input dataport ‘message_text’. The boolean type input dataport ‘abort_on_quit’ defines the states behaviour on canceling the dialog. If True, the state will return with the ‘abortion’ outcome, otherwise it just will return with ‘success’.

5.7.2 2ButtonDialog

This dialog features the same text option as the one above, but lets you define two buttons via the input ports ‘option 1’ and ‘option 2’. Clicking the first button results in exiting with outcome ‘option_1’, hitting the second one returns the outcome ‘option_2’.

5.7.3 GenericButtonDialog

This dialog equals the 2ButtonDialog state, except that it lets you define more than two buttons. On clicking a button, the dialog will always exit with outcome ‘responded’ but puts the index of the clicked button in the output dataport ‘response_id’.

5.7.4 InputDialog

This dialog contains two buttons like the 2ButtonDialog but also features a ‘message_text’ entry field and an optional ‘checkbox_text’ entry field, which could be used for a ‘remember’ option or something similar. The checkbox is only placed if a string is present for the ‘checkbox_text’ input dataport. The checkbox state is written to the boolean output dataport ‘checkbox_state’, the entered text to ‘entered_text’.

5.7.5 ColumnCheckboxDialog

This dialog contains buttons like the 2ButtonDialog but also features a single column of checkboxes with labels attached to them. These labels are defined via the ‘checkbox_texts’ input dataport as a list of strings. The states of those checkboxes are emitted as a bool list via the ‘checkbox_states’ output data port. A checked checkbox returns ‘True’.

**CHAPTER
SIX**

BUILDING THE DOCUMENTATION

The documentation is written in the reStructuredText markup language and resides in the *doc* folder. It contains instructions to parse the source code to also include the documentation about the API.

If you want to build the documentation as HTML page, you have to run *sphinx*. This can either be done with the provided PyCharm run configuration *Build Documentation* or manually in the console:

```
$ cd /path/to/rafcon/repository
$ sphinx-build -b html doc build_doc
```

This will build the documentation in the *build_doc* folder. Pass *-b pdf* to generate a PDF instead of a HTML page.

If you want to clean the RAFCON directory */install/directory/rafcon* from any build/installation artifacts, you can do so with:

```
cd /install/directory/rafcon
rm -r build/ build_doc/ .eggs/ .cache/
```


CONFIGURATION

RAFCON can be configured using two config files, one for the core and one for the GUI. The config files are automatically generated (if not existing) on the first run of *RAFCON*. It is stored in your home folder: `~/.config/rafcon/` with name `config.yaml` and `gui_config.yaml`, respectively. The path can be changed when running the `start.py` script with argument “`-c`”. The syntax used is [YAML](#).

7.1 Core Configuration

Example:

A typical config file looks like this:

```
TYPE: SM_CONFIG

LIBRARY_PATHS: {
    "generic": "${RAFCON_LIB_PATH}/generic",
    "tutorials": "${RAFCON_LIB_PATH}../examples/tutorials",
    "ros": "${RAFCON_LIB_PATH}../examples/libraries/ros_libraries",
    "turtle_libraries": "${RAFCON_LIB_PATH}../examples/libraries/turtle_libraries",
    "intermediate_level": "${RAFCON_LIB_PATH}../examples/functionality_examples"
}
LIBRARY_RECOVERY_MODE: False
LOAD_SM_WITH_CHECKS: True

STORAGE_PATH_WITH_STATE_NAME: True
MAX_LENGTH_FOR_STATE_NAME_IN_STORAGE_PATH: None
NO_PROGRAMMATIC_CHANGE_OF_LIBRARY_STATES_PERFORMED: False

IN_MEMORY_EXECUTION_HISTORY_ENABLE: True
FILE_SYSTEM_EXECUTION_HISTORY_ENABLE: True
EXECUTION_LOG_PATH: "%RAFCON_TEMP_PATH_BASE/execution_logs"
EXECUTION_LOG_SET_READ_AND_WRITABLE_FOR_ALL: False

SCRIPT_RECOMPILATION_ON_STATE_EXECUTION: True
SCRIPT_COMPILE_ON_FILESYSTEM_LOAD: True
```

Documentation:

In the following, all possible parameters are described, together with their default value:

TYPE

Type: String-constant

Default: SM_CONFIG

Specifying the type of configuration. Must be SM_CONFIG for the core config file.

LIBRARY_PATHS

Type: Dictionary with type(key) = String and type(value) = String

Default: {"generic": "\${RAFCON_LIB_PATH}/generic"}

A dictionary holding all libraries accessible in RAFCON. The key of the dictionary is a unique library identifier. This unique identifier will be used as library name, shown as root of the library hierarchy in the library tree. The value of the dictionary is a relative or absolute path on the file system that is searched for libraries. Relative paths are assumed to be relative to the config file. Environment variables are also allowed.

LIBRARY_RECOVERY_MODE

Type: boolean

Default: False

If this flag is activated, state machine with consistency errors concerning their data ports can be loaded. Instead of raising exceptions only errors are printed. Invalid transitions and data-flows will just be removed. This mode can be used to fix erroneous state machines. Intermediate and expert users can also keep this setting enabled all the time.

LOAD_SM_WITH_CHECKS

Type: boolean

Default: True

If this flag is activated, every state is checked for consistency before loaded. If set to false all consistency checks will be skipped. This leads to much faster loading times. However, if there are consistency errors RAFCON tries to open the state machines and will fail.

STORAGE_PATH_WITH_STATE_NAME

Type: boolean

Default: True

If set to True the paths to save states will contain the state names. If False only the state IDs will be used to create the storage path.

MAX_LENGTH_FOR_STATE_NAME_IN_STORAGE_PATH

Default: None

Unit: number

Specifies the maximum length of a state name in the storage path. If the state name is longer than the specified value, the state name is truncated. If the value is set to None the whole state name is used inside the path.

NO_PROGRAMMATIC_CHANGE_OF_LIBRARY_STATES_PERFORMED

Type: boolean

Default: False

Set this to True if you can make sure that the interface of library states is not programmatically changed anywhere inside your state machines. This will speed up loading of libraries.

EXECUTION_HISTORY_ENABLE

Type: boolean

Default: True

Enables execution history. The execution history is required for backward execution and execution logging to the file system.

EXECUTION_LOG_ENABLE

Type: boolean

Default: True

Enables the logging of rafcon execution histories to the file system. Every time a statemachine is executed, a python shelve is created in the execution log directory, e.g. `/tmp/rafcon_execution_logs/rafcon_execution_log_99-Bottles-of-Beer_2017-08-31-16-07-17.shelve`. Some helpful utility functions for working with log files through python are in: `import rafcon.utils.execution_log`. A tiny tiny code snippet which shows how to use the pandas.DataFrame representation to query the outcomes of a state named 'CheckFinished' is here:

<https://rsrc-github.robotic.dlr.de/common/rafcon/pull/324#issuecomment-2520>

EXECUTION_LOG_PATH:

Type: String

Default: `"/tmp/"`

Sets the target path of the execution logs

EXECUTION_LOG_SET_READ_AND_WRITABLE_FOR_ALL:

Type: boolean

Default: False

If True, the file permissions of the log file are set such that all users have read access to this file.

SCRIPT_RECOMPILATION_ON_STATE_EXECUTION:

Type: boolean

Default: True

If True, the script of an `ExecutionState` will be recompiled each time the state is executed, effectively resetting all global variables. For reasons of backwards compatibility, the default value is True. It is recommended to set the value to False, causing a recompilation only when the execution of a state machine is newly started, which is a bit faster and allows to share data between consecutive state executions.

SCRIPT_COMPILE_ON_FILESYSTEM_LOAD:

Type: boolean

Default: True

If True, the script of an `ExecutionState` will be recompiled each time the state is loaded from file-system. For faster loading times the setting can be changed to false. Then, however, it might be the case that during runtime, script compilation error occur.

7.2 GUI Configuration

A typical config file looks like this:

```
TYPE: GUI_CONFIG

SOURCE_EDITOR_STYLE: rafcon

GAPHAS_EDITOR_AUTO_FOCUS_OF_ROOT_STATE: True
ENABLE_CACHING: True
THEME_DARK VARIANT: True
DRAG_N_DROP_WITH_FOCUS: False

WAYPOINT_SNAP_ANGLE: 45
WAYPOINT_SNAP_MAX_DIFF_ANGLE: 10
WAYPOINT_SNAP_MAX_DIFF_PIXEL: 50
```

(continues on next page)

(continued from previous page)

```
PORT_SNAP_DISTANCE: 5

LOGGING_SHOW_VERBOSE: False
LOGGING_SHOW_DEBUG: False
LOGGING_SHOW_INFO: True
LOGGING_SHOW_WARNING: True
LOGGING_SHOW_ERROR: True
CONSOLE_FOLLOW_LOGGING: True

LIBRARY_TREE_PATH_HUMAN_READABLE: False
SUBSTITUTE_STATE_KEEPES_STATE_NAME: True

MINIMUM_SIZE_FOR_CONTENT: 30
MAX_VISIBLE_LIBRARY_HIERARCHY: 2
NO_FULLY_RECURSIVE_LIBRARY_MODEL: True

USE_ICONS_AS_TAB_LABELS: True

SHOW_NAMES_ON_DATA_FLOWS: True
SHOW_CONTENT_LIBRARY_NAME_TRANSPARENCY: 0.5
ROTATE_NAMES_ON_CONNECTIONS: False

HISTORY_ENABLED: True

KEEP_ONLY_STICKY_STATES_OPEN: True

AUTO_BACKUP_ENABLED: True
AUTO_BACKUP_ONLY_FIX_FORCED_INTERVAL: False
AUTO_BACKUP_FORCED_STORAGE_INTERVAL: 120
AUTO_BACKUP_DYNAMIC_STORAGE_INTERVAL: 20
AUTO_RECOVERY_CHECK: False
AUTO_RECOVERY_LOCK_ENABLED: False

SESSION_RESTORE_ENABLED: True

NUMBER_OF_RECENT_OPENED_STATE_MACHINES_STORED: 20

AUTO_APPLY_SOURCE_CODE_CHANGES: True

CHECK_PYTHON_FILES_WITH_PYLINT: False

DEFAULT_EXTERNAL_EDITOR:
PREFER_EXTERNAL_EDITOR: False

RESTORE_UNDOCKED_SIDEbars: True

FULLSCREEN_SHOW_TOOLBAR: True

NOTIFICATIONS_MINIMUM_LOG_LEVEL: 30
NOTIFICATIONS_DURATION: 3
```

(continues on next page)

(continued from previous page)

```

STATE_SELECTION_INSIDE_LIBRARY_STATE_ENABLED: True
LIBRARY_TREE_TOOLTIP_INCLUDES_ROOT_STATE_DESCRIPTION: True

ZOOM_WITH_CTRL: False

SEMANTIC_DATA_MODE: False
SHOW_PATH_NAMES_IN_EXECUTION_HISTORY: False
EXECUTION_TICKER_ENABLED: True
EXECUTION_TICKER_PATH_DEPTH: 3

# 300 is equal to glib.PRIORITY_LOW which is lower than the default gtk priority
LOGGING_CONSOLE_GTK_PRIORITY: 300

SHORTCUTS:
abort: Escape
add: <Control>A
add_execution_state: <Alt>E
add_hierarchy_state:
- <Alt>H
- <Control><Shift>A
add_preemptive_state: <Alt>C
add_barrier_state: <Alt>B
add_output: <Alt>U
add_input: <Alt>N
add_outcome: <Alt>T
add_scoped_variable: <Alt>V
apply: <Control><Shift>E
backward_step: F9
close: <Control>W
copy: <Control>C
cut: <Control>X
data_flow_mode: <Control><Shift>D
delete: Delete
down:
- <Control>Down
- <Control><Shift>Down
fit: <Control>space
group: <Control>G
info: <Control>I
is_start_state:
- <Control>E
- <Control><Shift>X
transition_from_closest_sibling_state: <Control><Shift>C
transition_to_closest_sibling_state: <Control><Shift>V
transition_to_parent_state: <Control><Shift>B
left:
- <Control>Left
- <Control><Shift>Left
new: <Control>N
open: <Control>O
open_external_editor: <Control><Shift>Q
open_library_state_separately: <Control><Shift>space

```

(continues on next page)

(continued from previous page)

```
paste: <Control>V
pause: F7
quit: <Control>Q
redo:
- <Control>Y
- <Control><Shift>Z
reload: <Shift>F5
rename: F2
right:
- <Control>Right
- <Control><Shift>Right
run_to_selected: <Control><Shift>R
save: <Control>S
save_as: <Control><Shift>S
save_as_copy: <Control><Shift><Alt>S
save_state_as: <Control><Alt>S
substitute_state: <Control><Shift><Alt>S
show_aborted_preempted: <Control>P
show_data_flows: <Control>D
show_data_values: <Control>L
start: F5
start_from_selected: <Control>R
step: F4
step_mode: F6
stop: F8
undo: <Control>Z
ungroup:
- <Control><Shift>G
- <Control>U
up:
- <Control>Up
- <Control><Shift>Up
fullscreen: F11
```

Documentation:

TYPE

Type: String-constant

Default: GUI_CONFIG

Specifying the type of configuration. Must be GUI_CONFIG for the GUI config file.

SOURCE_EDITOR_STYLE

Type: string

Default: rafcon

The gtk source view style used in the script editor. Note: You can download different styles [here](#). The scripts have to be downloaded to <rafcon package directory>/share/gtksourceview-2.0/styles. “rafcon” is a style created to fit to the design of RAFCON.

GAPHAS_EDITOR_AUTO_FOCUS_OF_ROOT_STATE

Type: boolean

Default: True

If RAFCON is started with the Gaphas editor enabled this flag enables an initial auto focus of the root state after opening the state machine. If you do not like this feature simply disable it (False).

ENABLE_CACHING:

Default: True

Enables a accelerating caching feature.

THEME_DARK_VARIANT:

Default: True

If True, a dark theme will be used, else a light theme

PORT_SNAP_DISTANCE

Default: 5

Unit: Pixel

Maximum distance to a port, at which the moved end of a connection is snapped to a port (outcome, input, output, scoped variable).

LOGGING_SHOW_VERBOSE

Type: boolean

Default: False

The flag decides to activate the VERBOSE log level in the logging console view.

LOGGING_SHOW_DEBUG

Type: boolean

Default: False

The flag decides to activate the DEBUG log level in the logging console view.

LOGGING_SHOW_INFO

Type: boolean

Default: True

The flag decides to activate the INFO log level in the logging console view.

LOGGING_SHOW_WARNING

Type: boolean

Default: True

The flag decides to activate the WARNING log level in the logging console view.

LOGGING_SHOW_ERROR

Type: boolean

Default: True

The flag decides to activate the ERROR log level in the logging console view.

CONSOLE_FOLLOW_LOGGING

Type: boolean

Default: True

The flag decides to activate the follow mode in the logging console view and to stay on the last printed logger message.

LIBRARY_TREE_PATH_HUMAN_READABLE

Type: boolean

Default: False

The flag is substituting underscores with spaces in the library tree. Thereby it is thought for people who do not like spaces in file system paths but don't wanna have underscores in the library tree.

SUBSTITUTE_STATE_KEEPSTATE_NAME

Type: boolean

Default: True

The flag describes the default behavior of the substitute state action concerning the previous state name and the state name after the substitution. In the dialogs this can be adapted for each single operation via a check box. If the flag is True the name is taken from the original state. If the flag is False the name is taken from the state machine that substitutes the original state.

MINIMUM_SIZE_FOR_CONTENT

Default: 30

Unit: Pixel

Minimum side length (width and height) for container states to have their content (child states, transitions, etc.) shown. Currently only used in the old editor (OpenGL).

MAX_VISIBLE_LIBRARY_HIERARCHY

Default: 2

Number of hierarchy levels to be shown within a library state. High values cause the GUI to lag.

NO_FULLY_RECURSIVE_LIBRARY_MODEL

Type: boolean

Default: True

If True, GUI models are only loaded up to the MAX_VISIBLE_LIBRARY_HIERARCHY. Setting this to False will drastically increase the time for loading a state machine.

USE_ICONS_AS_TAB_LABELS

Type: boolean

Default: True

If True, only icons will be shown in the tabs of the notebooks of the left and right pane. Otherwise the text of the notebook tab is shown as text.

SHOW_NAMES_ON_DATA_FLOWS

Type: boolean

Default: True

If False, data flow labels will not be shown (helpful if there are many data flows)

SHOW_CONTENT_LIBRARY_NAME_TRANSPARENCY

Type: float

Default: 0.5

Set to a value between 0 and 1. Defines the transparency of the name of a LibraryState in the graphical editor, of which the content is shown.

ROTATE_NAMES_ON_CONNECTIONS

Type: boolean

Default: False

If True, connection labels will be parallel to the connection. Otherwise, they are horizontally aligned.

HISTORY_ENABLED

Type: boolean

Default: True

If True, an edit history will be created, allowing for undo and redo operations.

KEEP_ONLY_STICKY_STATES_OPEN

Type: boolean

Default: True

If True, only the currently selected state and sticky states are open in the “states editor” on the right side. Thus, a newly selected state closes the old one. If False, all states remain open, if they are not actively closed.

AUTO_BACKUP_ENABLED

Type: boolean

Default: True

If True, the auto backup is enabled. If False, the auto-backup is disabled.

AUTO_BACKUP_ONLY_FIX_FORCED_INTERVAL

Type: boolean

Default: False

If True, the auto backup is performed according to a fixed time interval which is defined by AUTO_BACKUP_FORCED_STORAGE_INTERVAL. If False, the auto-backup is performed dynamically according to AUTO_BACKUP_DYNAMIC_STORAGE_INTERVAL. This means that RAFCON tries to avoid user disturbances by waiting for the case that the user does not perform any changes to the state machine for AUTO_BACKUP_DYNAMIC_STORAGE_INTERVAL seconds. If this happens RAFCON will perform a backup. Still AUTO_BACKUP_FORCED_STORAGE_INTERVAL is used as a hard storage interval. More information about this can be found on [Auto Backup](#)

AUTO_BACKUP_FORCED_STORAGE_INTERVAL

Default: 120

Unit: Seconds

Time horizon for a forced auto-backup if AUTO_BACKUP_ONLY_FIX_FORCED_INTERVAL is True.

AUTO_BACKUP_DYNAMIC_STORAGE_INTERVAL

Default: 20

Unit: Seconds

Time horizon after which the auto-backup is triggered if there was no modification to the state-machine for a time interval of this size. (only if AUTO_BACKUP_ONLY_FIX_FORCED_INTERVAL is False)

AUTO_RECOVERY_CHECK

Default: False

If True, the auto back module will check for backups of crashed RAFCON instances. This comfortable feature only can be used if the crashed instances or state machines were already created with AUTO_RECOVERY_LOCK_ENABLED and AUTO_BACKUP_ENABLED set to True.

AUTO_RECOVERY_LOCK_ENABLED:

Default: False

If True, the auto backup will put lock-files into the respective backup folder to label not correctly/cleanly closed state machines and instances. The auto recovery check is searching for these lock-files.

SESSION_RESTORE_ENABLED:

Default: True

If True the current session is stored into the runtime configuration and restored after restarting RAFCON.

NUMBER_OF_RECENT_OPENED_STATE_MACHINES_STORED:

default: 20

Maximum number of state machines that can be restored in a session.

AUTO_APPLY_SOURCE_CODE_CHANGES

Default: True

If True, RAFCON will apply source code changes on saving a state machine.

CHECK_PYTHON_FILES_WITH_PYLINT

Default: False

If True, RAFCON checks the script file with pylint before saving it. In case of an error a message dialog will pop up to warn the user about the error.

DEFAULT_EXTERNAL_EDITOR

Default: Empty

Holds the command for the editor to open the script.py file with, if the user clicks the ‘Open externally’ button in the source editor window. The command can be anything and results in a shell command with the following pattern: ‘<DEFAULT_EXTERNAL_EDITOR> script.py’.

PREFER_EXTERNAL_EDITOR

Default: False

If True, RAFCON will assume that the user always wants to work with a different editor than the internal one. If the ‘Open externally’ button is clicked, the source text is locked the whole time and a ‘Reload’ button reloads the saved file into RAFCON. If False, it is recommended to close the externally opened script.py everytime you are done editing.

RESTORE_UNDOCKED_SIDEbars

Default: True

If True, RAFCON will restore undocked windows from the last RAFCON-instance run.

FULLSCREEN_SHOW_TOOLBAR

Default: True

If True, the toolbar with execution and state buttons is shown in fullscreen mode.

NOTIFICATIONS_MINIMUM_LOG_LEVEL

Default: 30

Minimum log level of messages that shell show up in the notification bar. 40 corresponds to ERROR, 30 to WARNING, 20 to INFO, 10 to DEBUG and 5 to VERBOSE. If this is set to a level higher than 40, no notifications are shown.

NOTIFICATIONS_DURATION: 3

Default: 3

Number of seconds a notification is shown. If set to 0, the notification must be closed manually.

STATE_SELECTION_INSIDE_LIBRARY_STATE_ENABLED:

Default: True

If set to True, states inside library states can be selected.

LIBRARY_TREE_TOOLTIP_INCLUDES_ROOT_STATE_DESCRIPTION:

Default: True

If set to True, tooltip include the root state description text if the hovered library tree element (leaf element) is a real state machine.

ZOOM_WITH_CTRL:

Default: False

If set to True the user has to press the CTRL button to zoom into a state machine.

SEMANTIC_DATA_MODE

Default: False

If True, RAFCON gives the semantic data editor of each state more vertical space. The vertical space is taken from the port/connection widget. This is especially useful, when working a lot with semantic data.

SHOW_PATH_NAMES_IN_EXECUTION_HISTORY

Default: False

If True, RAFCON shows the state paths next to the state names in each execution history entry.

EXECUTION_TICKER_ENABLED

Default: True

If True, the execution ticker will prompt activity into respective widget.

EXECUTION_TICKER_PATH_DEPTH

Default: 3

Number of state names shown in active path (by names) starting from the lowest leaf state as the last and cutting away the first and following if to much.

LOGGING_CONSOLE_GTK_PRIORITY:

Default: 300

Unit: Priority

Sets the priority of logging anything to the console widget. The lower the number, the higher the priority. If the priority is too high, than the GUI will lag during execution, as the console widget will then slow down the rendering of gaphas / OpenGL

SHORTCUTS

Type: dict

Default: see example `gui_config.yaml` above

Defines the shortcuts of the GUI. The key describes the action triggered by the shortcut, the value defines the shortcut(s). There can be more than one shortcut registered for one action. See [GTK Documentation](#) about more information about the shortcut parser. Not all actions are implemented, yet. Some actions are global within the GUI (such as ‘save’), some are widget dependent (such as ‘add’).

7.3 Environment variables

Next to the configuration files, a number of environment variables exist that allow for further configuration.

7.3.1 RAFCON_LOGGING_CONF

See *Logging configuration*.

7.3.2 RAFCON_LIBRARY_PATH

An alternative option to specify your RAFCON libraries, which can e.g. be handy in combination with RMPM. See [Option 2](#).

7.3.3 RAFCON_PLUGIN_PATH

Use this variable to specify the RAFCON plugins that are to be loaded. See [Plugin Interface](#).

7.3.4 RAFCON_START_MINIMIZED

If the env variable RAFCON_START_MINIMIZED is set (i.e., has a value which is not an empty string), RAFCON is started minimized/iconified. This comes in handy, when the tests are run. You can then continue working, without RAFCON windows repeatedly being opened and closed in the foreground.

7.4 Logging configuration

RAFCON uses the default Python logging package for logging. Starting with version 0.9.7, logging handlers, filters, formatting and more can be configured using a JSON file. The default configuration can be found in `source/rafcon/logging.conf`. The configuration can be overwritten with a custom JSON file. To do so, specify the path to your configuration in the env variable `RAFCON_LOGGING_CONF`. For information about the logging package, please check the [official documentation](#).

Example:

To not destroy the behavior of RAFCON, the default configuration should be used as basis for your extensions. The following example shows how to add another logging handler, writing all messages to a file:

```
{  
    "loggers": {  
        "rafcon": {  
            "handlers": ["stdout", "stderr", "loggingView", "file"]  
        }  
    },  
  
    "handlers": {  
        "file": {  
            "class": "logging.handlers.RotatingFileHandler",  
            "formatter": "default",  
            "filename": "/tmp/rafcon.log",  
            "maxBytes": 1024,  
            "backupCount": 3  
        }  
    },  
}
```

7.5 Monitoring plugin configuration

The config file of the monitoring plugin contains all parameters and settings for communication. It is additionally needed next to the `config.yaml` and the `gui_config.yaml` to run the plugin. If it does not exist, it will be automatically generated by the first start of the `start.py` and stored at `~/.config/rafcon` as `network_config.yaml`. The path of the used config file can be changed by launching the `start.py` script with argument “-nc”.

Example:

The default `network_config.yaml` looks like:

```
TYPE: NETWORK_CONFIG
ENABLED: true
HASH_LENGTH: 8
HISTORY_LENGTH: 1000
MAX_TIME_WAITING_BETWEEN_CONNECTION_TRY_OUTS: 3.0
MAX_TIME_WAITING_FOR_ACKNOWLEDGEMENTS: 1.0
SALT_LENGTH: 6
SERVER: true
SERVER_IP: 127.0.0.1
SERVER_UDP_PORT: 9999
TIME_BETWEEN_BURSTS: 0.01
BURST_NUMBER: 1
CLIENT_UDP_PORT: 7777
```

Documentation:

TYPE

Type: string

Default: NETWORK_CONFIG

Specifying the type of configuration. Must be NETWORK_CONFIG for the network config file.

ENABLED

Type: boolean

Default: True

The monitoring plugin is only used if this value is set to True.

HASH_LENGTH

Type: int

Default: 8

If you have many different message contents, increase this number.

HISTORY_LENGTH

Type: int

Default: 1000

MAX_TIME_WAITING_BETWEEN_CONNECTION_TRY_OUTS

Type: float

Default: 3.0

MAX_TIME_WAITING_FOR_ACKNOWLEDGEMENTS

Type: float

Default: 1.0

Maximum waiting time for an acknowledgement after sending a message which expects one.

SALT_LENGTH

Type: int

Default: 6

SERVER

Type: boolean

Default: True

Defines if the RAFCON instance should start as server or client. If `False` process will start as client.

SERVER_IP

Type: string

Default: `127.0.0.1`

If RAFCON is started as client, SERVER_IP contains the IP to connect to.

SERVER_UDP_PORT

Type: int

Default: 9999

Contains the UDP port of the server which shall be connected to.

TIME_BETWEEN_BURSTS

Type: float

Default: `0.01`

Time between burst messages (refer to BURST_NUMBER).

BURST_NUMBER

Type: int

Default: 1

Amount of messages with the same content which shall be send to ensure the communication.

CLIENT_UDP_PORT

Type: int

Default: 7777

Contains the UDP port of the client

AUTO BACKUP

There are session restore and auto backup features on state machine level which are enabled by default and which can be used to backup and restore your work. These features can be enabled/disabled by the parameter SESSION_RESTORE_ENABLED and AUTO_BACKUP_ENABLED in the GUI config.

The session restore uses the backup of state machines and other GUI information to restore your open state machine tabs and respective selection situation.

If the state machine auto backup is enabled RAFCON creates temporary backups of your open state machines. You can find these after a crash of RAFCON on your computer in \$RUNTIME_BACKUP_PATH = /tmp/rafcon-\$USER/\$PID/runtime_backup/. \$USER is your user name and \$PID was/is the process id of your RAFCON instance. If a state machine hasn't been saved before, it will be located at \$RUNTIME_BACKUP_PATH/not_stored_\$SM_ID, whereby \$SM_ID is the ID of the state machine. If your state-machine has already been stored, the state machine backup path is \$RUNTIME_BACKUP_PATH/\$SM_BASE_PATH, whereas \$SM_BASE_PATH is the path to your state machine.

The automatic backup can either be disabled or enabled. If enabled either a fixed forced or dynamic interval can be set. Respective parameters are described in *Configuration* and start with AUTO_BACKUP_*.

In case of a fixed forced interval it is checked every duration T if there was a change to the state-machine. This means that a modification can maximal not been backuped for T. T is specified by AUTO_BACKUP_FORCED_STORAGE_INTERVAL.

In case of dynamic auto-backup it is tried to avoid user disturbances by waiting for a time-interval T*, within which the user has not modified the state-machine, to trigger the auto-backup while still using T as a hard limit. This means that a modification is possibly backup-ed every T* and forced after T. T* is specified by the config value AUTO_BACKUP_DYNAMIC_STORAGE_INTERVAL.

8.1 Auto Recovery

With the release 0.7.5 lock files for state machines and rafcon instances are introduced in the \$RUNTIME_BACKUP_PATH. State machines which were not proper stored can be identified and recovered on a formal way (dialog window) if the parameters AUTO_RECOVERY_LOCK_ENABLED and AUTO_RECOVERY_CHECK are set to True. By default those parameters are set False. In more detail AUTO_RECOVERY_LOCK_ENABLED results in the creation of a lock file and AUTO_RECOVERY_CHECK=True triggers a check on lock files for the whole /tmp/rafcon-\$USER folder and will offer optional recovery of respective state machines. So it is possible to enable lock file generation and only enable the check on lock files if explicitly needed.

An auto recovery of whole crashed sessions including their open state machines and tab states is currently not supported. Nevertheless, sessions can be restored if RAFCON was closed correctly and the config value SESSION_RESTORE_ENABLED is set to True.

GUI GUIDE

The following shows a common guide for the *RAFCON*-GUI. The page contains advises about how to use the widgets effectively.

9.1 Left Pane

9.1.1 Library Tree widget

The Library tree shows all libraries mounted via the LIBRARY_PATH defined in your configuration file, see [Configuration](#). Sub-library paths unfold and fold by double-click. Double-clicking on a library opens the library in the graphical editor. A right-click on the row of a library element opens a menu with the options “Add as Library”, “Add as template”, “Open” and “Open and run”. “Add as template” pastes a fully editable copy of the respective library into the actual selected state (as long as selected state is a ContainerState).

The Library Tree can be reloaded by pressing the the “Refresh Libraries”-Button in the Menu-Bar or Tool-Bar.

9.1.2 State Machine Tree widget

The State Machine Tree (or “State Tree”) shows all states that are in the actual selected state-machine, its state_id and the type of the state. A state is selected if the respective row is selected and the row of a state becomes selected if the state becomes selected by another widget e.g. the graphical editor. A selected state can be removed by pressing the delete-shortcut or a state can be added into a selected state/row by pressing the add-shortcut. See [Configuration](#) to define those shortcuts.

9.1.3 Global Variable Manager widget

The Global Variable Widget shows all existing global variables, their names, values and if they are locked. The name and value can be edited by clicking into the respective fields and “shift-tab”, “tab”, and the “arrow-keys” can be used to move throw the editable columns and variables. Press Enter or move by using “shift-tab” or “tab” to confirm the modification of a specific element. A selected global variable can be removed by pressing the delete-shortcut or a variable can be added by pressing the add-shortcut. See [Configuration](#) to define those shortcuts.

9.1.4 Modification History widget

The history is implemented as a tree of actions. Therefore a trail history view (list) and a branch history view (tree) are possible. The actual used representation is the branch history view, see figure.

In the top (below the label “HISTORY”) the edit history widget has three buttons for “Undo/Redo” of actions and “Reset” of the history. The reset deletes all history tree elements recorded so far. “Undo” and “Redo” also can be performed by using the shortcuts strg-z and strg-y as usual.

The view also can be modified by the disabling of the branch-history view (trail-history-view becomes active) using the check-button labeled “B” or by only enabling automatic folding of the not used branches by check-button “F”.

In the figure it can be seen that the trail history element with number 16 is selected (grey). Every number/id is one step of the state machine edit process that has been recorded. Action with the number 16 used the method “add transition” and connects two states with the IDs ZRMXSG and EOSVJL. Elements 17-18 are undone and thereby are labeled gray. All elements in the trail history from 0 to 16 are in use, so labeled white. All elements on a previous used branch but not active branch are also labeled grey, see Nr 11-12 and 3-4.

By selecting a specific element in the list/tree the version after this action is recovered. To jump to the initial situation of the state machine when the record of edit history started the user has to click element 0 which is labeled by action “None”.

HISTORY		
Nr	Action	Parameters
18	add state	ExecutionState with name 'Untitled' and id ZRMXSG, -1, EOSVJL, -1
17	add transition	ZRMXSG, 0, EOSVJL, 0
16	add transition	ZRMXSG, 0, EOSVJL, 0
15	change position	{'gui': {'editor_opengl': {'rel_pos': (3, 7141, 0, 0), 'x': 3, 'y': 7141}}, 'state': 'Untitled'}
14	add state	ExecutionState with name 'Untitled' and id PJBHUO, 0, XZIBDL, -2
b.13	change state type	ExecutionState with name 'Untitled' and id PJBHUO, 0, XZIBDL, -2
b.10	add outcome	success2
b.11	change root state type <class 'rafcon.statemachine.states.barrier.BoundaryState'>	<class 'rafcon.statemachine.states.barrier.BoundaryState'>
12	add state	ExecutionState with name 'Untitled' and id ZUKDFM, -2, XZIBDL, -2
9	add output data port	output_0, int, 0
8	add input data port	input_0, int, 0
7	name	Neww State
6	add transition	PJBHUO, 0, XZIBDL, -2
b.5	add state	ExecutionState with name 'Neww State' and id ZUKDFM, -2, XZIBDL, -2
b.2	add transition	ZUKDFM, -2, XZIBDL, -2
b.3	add state	ExecutionState with name 'Untitled' and id ZUKDFM, -2, XZIBDL, -2
4	add transition	ZUKDFM, -2, XZIBDL, -2
1	add state	ExecutionState with name 'Untitled' and id ZUKDFM, -2, XZIBDL, -2
0	None	

9.1.5 Execution History widget

In the execution history widget all runs of the currently selected state machine are visualized in a tree like structure. This can be used for debugging purposes as also all the context data of each executed state is visualized. The execution history can also be logged to disk. (see the logging variables in the *Core Configuration*)

9.2 Center Pane

9.2.1 Graphical State Machine Editor widget

There are two different graphical editors for editing a state machine. The current graphical editor uses the Python library “Gaphas”. The alternative graphical editor uses OpenGL for hardware acceleration with the graphics card. The *Configuration* explains how to switch between the two editors.

The Gaphas graphical editor is more advanced than the OpenGL one (and also much more pleasant to look at ;-)). There are some shortcuts, which you should be aware of:

- Zoom: Mouse scroll wheel
- Panning: Middle mouse button
- Move ports (along border of state): Ctrl + click and move cursor (select the state of the port beforehand)

- Move name of state: Ctrl + click and move cursor
- Resize state with content: Ctrl + click on corner handles
- Operations on the selected state: Right mouse button (opens context menu)
- Add new Execution State to selected state: Alt + E
- Add new Hierarchy State to selected state: Alt + H, Ctrl + Shift + A

You can also drag'n'drop states into the editor. This is possible from the four “+ *S” buttons below the editor and from the libraries widget.

9.2.2 Debug Console widget

The Debug Console can be found below the Graphical Editor. All messages will be displayed in it, whereas the type of the displayed messages can be filtered with the checkboxes on top of the console. As for the other widgets, the Debug Console can be unpinned by clicking the symbol in the upper right corner. A right-click into the console opens a menu providing the options to "Copy" selected output, "Select All" output or "Clear Logging View".

9.3 Right Pane (States Editor)

The right sidebar shows the “States Editor”. It can show several tabs, but by default, only the selected state is shown. However, you can *pin* a state tab by clicking on the needle icon within the tab.

The number within the tab shows the state machine id belonging to the state.

Every “State Editor” consists of the three widgets described below: The State Overview, State content (with widgets for the Source Editor, Logical Linkage and Data Linkage) and State Linkage Overview (with widgets for the Description and Semantic Data).

9.3.1 State Overview widget

The State Overview can be found directly under the “STATE EDITOR” headline. It provides the name of the selected state, which can be edited by clicking on it, and the fixed state ID. Additionally, the State Overview contains a dropdown menu, where the type of the state can be changed, and a checkbox which marks a state as start state.

9.3.2 Source Editor widget

The Source Editor is the first tab of the notebook in the middle. It is a editor with the three buttons “Apply” to save the file, “Cancel” to discard changes and “Open externally” to open the file in an external editor.

9.3.3 Logical Linkage widget

By clicking the middle tab of the center notebook, the sub-widgets Outcomes and Transitions, which represent the logical linkage of a state, can be reached. In the Outcomes widget the outcomes of the selected state are listed. It consists of the columns “ID”, “Name” and “To-State”. If the “To-State” is a hierarchy state the “To-Outcome” of the “To-State” is also shown. Next to the obligatory IDs “0”, “-1” and “-2”, it is possible to append own outcomes by clicking the “Add” button. A click on the “Remove” button will delete the selected outcome.

The Transitions sub-widget lists the transitions between the selected state and its siblings (or parent). There are the four columns: “Source State”, “Source Outcome”, “Target State” and “Target Outcome”. With the buttons “Add” and “Remove”, additional transitions can be added and selected ones can be deleted.

9.3.4 Data Linkage widget

The Data Ports and Data Flows sub-widgets, which represent the data linkage of a state, can be accessed by clicking on the last tab of the middle notebook. Within the Data Ports sub-widget it is possible to change between “Input Ports” and “Output Ports”. The currently selected one is highlighted in blue. Input and output ports work like function parameters. They consist of a “Name”, a “Data Type” and a “Default Value”. A click on the button “New” appends a new port, the button “Delete” removes the selected port. If the selected state is a container state, then also a tab for “Scoped Variable” emerges.

Similar to the transitions, the data flows of a state are shown in the lower part.

9.3.5 Data and Logical Linkage widget

A quick overview of all data ports and outcomes of the selected state.

9.3.6 State Description widget

The State Description sub-widget can be reached by clicking the second tab of the lower notebook. It is an editor, where comments or a description can be placed.

9.3.7 Semantic Editor widget

In this widget the semantic data of a state is shown and can be edited in a tree like data structure.

PLUGINS

RAFCON features a nice plugin concept and there are already some plugins out there. Most notable is the “Monitoring Plugin”, which supports remote access and observation of a state-machine running somewhere in the network.

In general there are three way of how plugins can interface *RAFCON*:

- Add or modify functionality by using predefined hooks
- Derive a *RAFCON* class and substitute the original one
- Use observers to add or modify specific behavior
- Monkey patch *RAFCON* functionality (if absolutely necessary)

There is a plugin template demonstrating how to implement and use a plugin (in `share/examples/plugins/templates`). Examples for more advanced usages of these methods can be found in the plugins introduced below.

10.1 Plugin Interface

The path of every *RAFCON* plugin has to be registered in the environmental variable `RAFCON_PLUGIN_PATH`. In the registered path, the `__init__.py` and the `hooks.py` of respective plugin should be situated. In `hooks.py`, the following functions (`hooks`) can be implemented.

10.1.1 `pre_init`

The function is called after all of *RAFCON* imports occurred and the *RAFCON* singletons have been created. In this function, *RAFCON* classes can be extended or completely substituted. Anyway, it is good to avoid monkey patches or big substitutions of classes at this point. Especially if you extend or substitute a class that is used in a singleton make sure that your change reaches all parts of *RAFCON* (by explicitly substituting objects). An example is given within the Execution Hook plugin.

10.1.2 `post_init`

The function is called after the command line parameters have been processed and everything is initialized (including the loading of state machines) Furthermore, the GUI and models are fully initiated (the observers of the GUI are registered to the core-objects and other observables). In this function, observers should register their observables. Simple examples can be found in the Execution Hook plugin and in the plugin template, a more complex example is given with the Monitoring plugin.

10.1.3 main_window_setup (GUI only)

The hook is called after the view has been registered, namely at the very end of the `register_view` methods of the `MainWindowController`. A reference to the main window controller is passed as an argument.

10.1.4 pre_destruction

When the GUI is running, this hook is called right before the `prepare_destruction` method of the Menu Bar is being called, thus before the GUI is being destroyed and closed. When only the core is in use, the hook is only called while the twisted reactor is running. The call is made before the twisted reactor is stopped.

10.1.5 post_destruction

When the GUI is running, this method is called after the GTK main loop has been terminated. Thus the main window has already been destroyed at this point. When only the core is in use, the hook is called right before the program ends.

10.2 Available plugins

For an overview of all available plugins have a look at [our website](#).

10.2.1 Monitoring Plugin

This RAFCON plugin enables monitoring RAFCON instances via unreliable UDP/IP. The RMPM package name is `rafcon_monitoring_plugin`. See also: [*Using the monitoring plugin*](#)

10.2.2 DDS Monitoring Plugin

This plugin facilitates monitoring and remote control of RAFCON state machines via RTI DDS.

Once the plugin has been set up, RAFCON can be started in either server or client mode. Currently, only one server per DDS domain is supported; if you would like to control multiple RAFCON instances you will need to separate them by using different domain IDs.

You can start RAFCON in server mode with or without GUI:

```
# Start server with GUI
rafcon --server
# Start server without GUI
rafcon_core --remote --server -o <path_to_state_machine> [<path_to_state_machine> ...]
```

In order to start RAFCON in client mode simply pass the `--client` parameter:

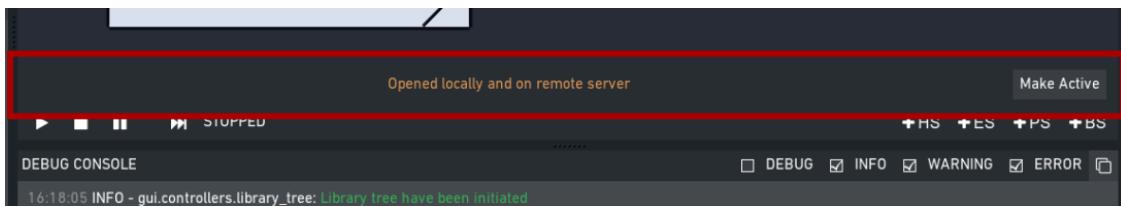
```
rafcon --client
```

There are no restrictions on the number of clients per domain, so you can connect as many clients as you wish.

When you start RAFCON in client mode and open a state machine you will notice a new information bar at the bottom:

This bar shows the status of the state machine on the remote server. There are four different states:

- **Opened locally** The state machine is opened locally on the client but not known to the remote server. It is therefore not possible to run this state machine.



- **Opened locally and on remote server** The state machine is opened both locally and on the remote server. In order to start this state machine on the server you will need to make it the *active state machine* by clicking on *Make Active*. This option is only available if the currently active state machine is not running or paused.
- **Active on remote server** The state machine is opened and active on the remote server. You can start it by using the common control options.
- **Running on remote server** The state machine is currently running on the remote server.

You can start the clients and server in any order; on startup, they will automatically retrieve/publish the current state. However, once the server quits, the information shown on the clients will be outdated (that is, they will show an active/running state machine even if there is no server running). Note that there is no “authoritative” client and the server will process the incoming commands simply in the order they arrive. Invalid commands will be dismissed.

10.2.3 Execution Hook Plugin

This RAFCON plugin enables to use execution hooks on changes in the execution engine. The RMPM package name is `rafcon_execution_hooks_plugin`. At the moment, the plugin only enables this hooks for the state-machine root state.

10.2.4 Plugin Template

The plugin template can be found in `[RAFCON root path]/share/examples/plugins/templates`. If you put this path into your `RAFCON_PLUGIN_PATH` environment variable, the plugin will be automatically loaded.

FAQ

On this page, we collect Frequently Ask Questions (FAQ) about *RAFCON*. At the moment, there are three categories of questions concerning *Core*, *API* or *GUI*. Some more answered questions can be found in our [list of closed issues](#) labeled “question”. If you have a new question, please [create an issue](#).

11.1 Core

Questions that concern the core functionality and state machine execution.

11.1.1 Where I can configure the RAFCON core?

The core RAFCON configuration file is per default situated here in `~/.config/rafcon/config.yaml`, but can also be passed to RAFCON using the command line parameter `-c /path/to/your/core/config.yaml`. For further explanation see [Configuration](#).

11.1.2 Where can instances of global objects be hold?

Global objects can be initialized at any point using the global variable manager. Storing such variables in the global variable manager (most of the time) only makes sense, when storing them per reference. Keep in mind: Global variables are ugly and make your state machine less modular. Use them as little as possible!

11.1.3 How does concurrency work?

A concurrency state executes all child states concurrently. A barrier concurrency state waits for all its children to finish their execution. A preemptive concurrency state only waits for the first state to finsih and preempts the others. Each state gets its own thread when it is called, so they run completely independently. Also each state gets its own memory space, i.e. all input data of the a state is copied at first and then passed to the state.

Direct interaction between two concurrent branches inside a concurrency state is not possible in RAFCON, except by using global variables (please use them with care, as heavily using them make your state machine less modular). The reasoning for this is that synchronization of many processes is a hot topic and very error prone. Our recommendation is:

1. Distribute your data to parallel branches
2. Do stuff in parallel
3. After concurrent branches finished: Collect data of branches and process it
4. Distribute data it again

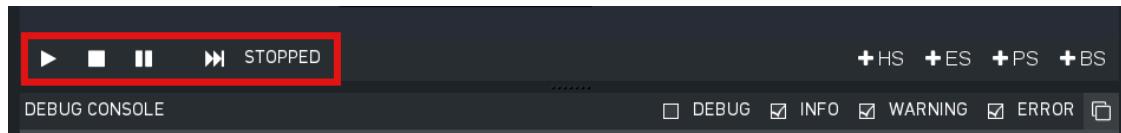
5. Do stuff in parallel etc.

Monitoring of different system states works via preemption and by using global variables as e.g. global signals. A separate resource manager and world model to hold the information about the system is definitely needed for more complex use cases. RAFCON is not a tool for holding complex data, so don't try to use it for this.

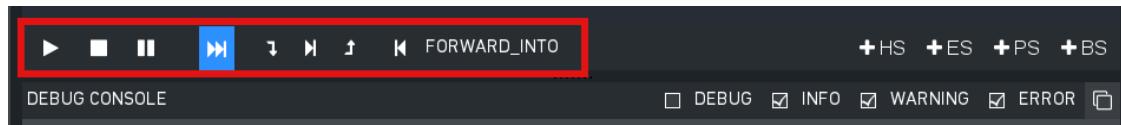
See also [How does preemption work? How do I implement preemptable states correctly?](#)

11.1.4 How does execution control – including stepping mode – work?

In the execution widget below the graphical editor, the execution of a state machine can be controlled.



Here the user can start, pause and stop the state machine. Furthermore, a step mode can be activated.



In the step mode, the user can trigger four kinds of step: "Step Into", "Step Over", "Step Out" and "Backward Step".

The "Step Into" simply executes the next state in the state machine. So the execution goes down and up the hierarchy.

The "Step Over" makes a step on the same hierarchy level, independent on how many sub-states the next state will trigger. If the execution reaches the end of the hierarchy it steps out to the hierarchy.

The "Step Out" executes all states in the current hierarchy until the execution reaches an outcome of the current hierarchy.

The "Backward Step" triggers a backward step with respect to the current execution history. Before and after the execution of each state all the context data (i.e. the scoped data) of the current hierarchy is stored. The scoped data includes all the data that was given to the current container state as input and that was created by the child states with their outputs. A backward step now loads all the scoped data which was valid after the execution of the state, executes the state in backward mode and then loads the scoped data which was valid before executing the state. Executing a state in backward mode means executing an optional `def backward_execute(self, inputs, outputs, gvm)` function. The inputs and outputs of the function are the input data of the state (defined by its data flows) loaded from the current scoped data. If the `backward_execute` function is not defined, nothing is executed at all. For an example backward-stepping state machine, have a look at the "functionality_examples" in the RAFCON Git repository: `[path_to_git_repo]/share/examples/functionality_examples`.

11.1.5 What does pause and stop do?

Pausing a state machine prevents the current state to "take" the next transition. Furthermore a paused-event is triggered for each state.

Stopping a state machine also prevents the current state to "take" the next transition. Instead of taking the transition selected by the stopped state, the execution runs the state connected to the "preempted" outcome of the stopped state. If no state is connected to the "preempted" outcome, the current state hierarchy is left with the "preempted" outcome. Stopping a state does not stop the thread of the state itself. It only triggers a preempted-event for each state.

For information on how to correctly listen to pause or preempted events inside a state, see [What happens if the state machine is paused? How can I pause running services, e. g. the robot?](#)

11.1.6 Stepping through a state machine is cool, but I need to step through the code within a state. Can I do this?

RAFCON has no special feature for this, but Python has. It is called [the Python Debugger](#). All you have to do is inserting the following line of code into your `execute()` method:

```
import pdb; pdb.set_trace()
```

Alternatively, you can also use [the IPython Debugger ipdb](#), giving you e.g. syntax highlighting and better tracebacks:

```
import ipdb; ipdb.set_trace()
```

If this line of code is hit, the execution pauses and you need to switch to the terminal where you started RAFCON. You can now, for example, inspect the parameters of the `execute()` method by entering `a`. This will output something like

```
inputs = {'counter': 3}
outputs = []
gvm = <rafcon.core.global_variable_manager.GlobalVariableManager object at
    ↪0x7ff6a6541090>
```

A good tutorial about the powerful possibilities you have within the debugger can be found on [Real Python](#).

11.1.7 How does preemption work? How do I implement preemptable states correctly?

Preemption is achieved in *preemptive concurrency states*. All direct children of these states are executed in parallel in separate threads. These direct children can be of all kinds of states: execution states, libraries or any type of container. The direct child, which finishes its execution first (by returning an outcome), causes all sibling states to stop (preempt). If all siblings have been preempted, the execution of the preemptive concurrency state is finished.

When a state is preempted, the preemption starts at the innermost running child state and propagates up: First, the preempted flag of the innermost running children is set to True. Then it is waited until the state returns an outcome. The outcome itself is ignored, as a preempted state is always left on the preempted outcome. If the preempted outcome is connected, the connected state is executed. Otherwise, the hierarchy is left and the parent state is preempted in the same way, until the preemptive concurrency state is reached.

States have the possibility to define an action to be executed when being preempted. This is intended e. g. for closing any open resources. For this, the user connects a state with the desired logic to the preempted outcome of the state opening the resource or to its parent. For direct children of a preemptive concurrency state, no preemption routine can be defined. In this case another hierarchy state has to be introduced.

Running states are only requested to preempt but are not and cannot be forced to preempt. This means that states should run as short as possible. If this is not feasible, the user has to ensure that a state is preemptable. If a state contains a loop, the user should check in each iteration, whether the flag `self.preempted` is True and stop in this case. If a state needs to pause, `self.preemptive_wait(time)` or `self.wait_for_interruption()` (see next question) should be used instead of `time.sleep(time)`. The former method is preempted if the state is urged to preempt, the latter is not. It returns True if the wait time wasn't reached, i. e. if the method was preempted before. If None (or nothing) is passed to `self.preemptive_wait(time)`, the method waits infinitely for being preempted. Note that preemption is not only caused by sibling states within a preemptive concurrency state, but states are also preempted if the execution of the whole state machine is stopped (by the user clicking "Stop").

This should also be kept in mind when developing libraries. As a user could use libraries in Preemptive Concurrency States, libraries should be designed in this way.

11.1.8 What happens if a state machine is paused? How can I pause running services, e. g. the robot?

The basic behavior is simple: If a state machine is paused, no more transition is being followed. I. e., if a state returns an outcome, the execution is stopped at this outcome. When the execution is resumed (by clicking the “Run” button), the execution continues at this outcome.

Yet, states are not forced to pause, just as for preemption. Only the flag `self.paused` is set. Therefore, states should be implemented with care, if they run for a longer time. For this, one can use two helper methods, `self.wait_for_interruption(timeout=None)` and `self.wait_for_unpause(timeout=None)`. Alternatively, one can directly access the Python `threading.Events` `self._started`, `self._paused`, `self._preempted`, `self._interrupted` and `self._unpaused`. The “interrupted” event is a combination of “paused” and “stopped”; “unpaused” is a combination of “started” and “stopped”. An example implementation can be seen in the following:

```
def execute(self, inputs, outputs, gvm):
    self.logger.info("Starting heartbeat")

    for _ in xrange(10):
        self.logger.info("pulse")
        self.wait_for_interruption(1)

        if self.preempted:
            return "preempted"
        if self.paused:
            self.logger.debug("Heart paused")
            self.wait_for_unpause()
            if self.preempted:
                return "preempted"
            self.logger.debug("Heart reanimated")
    return 0
```

An execution state with this code snippet would print “pulse” once per second (`self.wait_for_interruption(1)`). The wait command is interrupted, if either the user clicks “pause” or the state is preempted (state machine is stopped or a child state running in parallel in a preemptive concurrency state finishes). Therefore, the two event types are checked. If the state is to be preempted, the state follows that request (`if self.preempted: return "preempted"`). If the execution was paused, the state waits for a resume (`self.wait_for_unpause()`). The wait command is interrupted either by the continuation of the execution or by a complete stop of the execution. The former manifests in the `self.started` flag to be set, the latter by the set of the `self.preempted` flag.

If an external service is involved, e. g. for commanding a robot, that service might also be paused. For this, one can pass the respective services to the robot. This requires the external service to be written in Python.

11.1.9 How to handle a state abortion correctly?

As arbitrary python code is allowed in a state, the execution of a state can raise arbitrary python errors. If an error is raised the state is left via the “aborted” outcome. Furthermore the error of the state is stored and passed to the next state as an input port with the name “error”. The error (e.g. its type) can be checked and used for error handling mechanisms. If no state is connected to the “aborted” outcome of the aborted state, the error is propagated upwards in the hierarchy until a state is handling the abortion or the state machine is left. An example state machine on how to use such a error handling can look like is given in `$(RAFCON_GIT_REPO_PATH)/tests/assets/unit_test_state_machines/error_propagation_test`. If the error handling state is a hierarchy state the “error” input data port must be manually forwarded to the first child state i.e. a `input_data` port for the hierarchy and the child state has to be created and connected.

11.1.10 How does python-jsonconversion handle string types?

Serialized strings are stored in a file in ASCII encoding, but they are read from a file as unicode. Thus explicit conversions to ASCII has to done if the type of the string matters.

11.1.11 Why is RAFCON not event-based?

tl; dr: RAFCON was created to enable the development of goal-driven behavior, in contrast to reactive behavior (for which many frameworks rely on events). However, RAFCON can be used to implement reactive behaviors as well. Quite elegantly, using observers!

Long Answer: RAFCON state machines are no state machines in the classical sense. A state in RAFCON is not a state like in a FSM (i.e., a system state), but in which a certain action is executed. Thus, a state is rather related to flowchart block. The HPFD formalization underlying RAFCON is defined in (see <https://elib.dlr.de/112067/>). Related to robotics, a RAFCON state is a state in the robot's behavior (e.g., performing pick-up action) and not a robot's system state (battery at 20V, position at [x, y]).

We have employed RAFCON for many goal-driven scenarios (see <https://dlr-rm.github.io/RAFCON/projects>). In goal-driven scenarios (in contrast to reactive ones) all changes of the environment are an effect of the robot's own actions. There are only a few "external events" the robot has to react to (low voltage, bad signal etc.).

As stated before, RAFCON can not only be used to build goal-driven behavior but also to build reactive systems. Classical state machines or statecharts can be used to implement reactive behavior as well and they DO rely on events. Behavior trees are also a very powerful concept for reactive behavior and they do NOT rely on events. For RAFCON, we don't rely on events either. We employ observers to take care of events. To implement them, you have to use (preemptive) concurrency states. E.g. you can set up a preemptive concurrency state with 5 observers. The first one who fires decides the course of action. You can place different observers on different hierarchy levels. Thereby, you can define the scope of your 'event' and your 'event handler'.

We have a lot of experience with event-based state machines like boost statechart, due to our experience in the RoboCup competition (see https://link.springer.com/chapter/10.1007/978-3-642-39250-4_5). One lesson learned was that applying event-based state machines does not scale.

To understand this, imagine a classical use-case scenario:

- You are using a non-graphical library for programming event-based state machines like boost statechart.
- You have a hierarchical state machine with several hundreds of states inside several hierarchy layers.
- You have 30 types of events that are continuously firing onto your state machine.
- In certain states you react to some events.
- Most of the time you neglect 90% of the events and are only interested in some events defined by the current set of active states. Nevertheless, the events arbiter has to handle *every* event, which can be inefficient.

Now try to answer the following questions, that would be raised during runtime:

1. What is the set of currently active states?
2. What events do I currently listen to?
3. What is the context (hierarchy level, possible predecessors, possible successors) of each of those state?
4. Since when do I listen to event_x? Until which state will I listen to event_x?
5. I receive an event that I cannot process know. I defer it to a later point in time (*event deferral*).
 - How long are events valid (*event caching*)?
 - Another event with the same type arrives meanwhile. Should I keep the old event (*event expiration*)?

- Another event, which is more important arrives. Should I react to the new event and preempt the current event handling (*event prioritization*)?

It is obviously not trivial! Of course, you could answer those questions, but it is cumbersome and you quickly lose the overview.

Thus, we implemented a graphical user interface, where you can easily answer those questions:

1. You clearly see all currently executed states, highlighted in green in the graphical 2.5D layout and in your execution history tree!
2. Event observers are states. For active state, see answer 1.
3. You clearly see the context of each state visually drawn (if you want to have a closer look, simply zoom in)
4. Events map to observers, observers to states; their entry and exit is clearly visualized => see 3.
5. As we do use observers instead of events we don't have to care about all this complex topics using the limited power of events (Events are error-prone anyways. Adobe reported that 50% of all bugs are related to event handling: https://stlab.cc/legacy/figures/Boostcon_possible_future.pdf) Instead of complex event handling, you can create powerful observer structures also in a nested manner using hierarchies and outcome forwarding.

Concerning how to deal with the four mentioned event-challenges using observers:

- Event deferral: Use a proper hierarchical layout for your observers. The observer in a higher hierarchy layer will defer its events, until the execution of its child observers is finished.
- Event caching: As you have one observer per event, caching is done automatically until the observer is preempted.
- Event expiration: By preempting an observer sibling you clear the “cache” for this event. Make sure you have a proper hierarchical observer layout: i.e., if an observer must not clear the cache for a certain event, pull it's observer one hierarchy up!
- Event prioritization: Make sure you have a proper hierarchical observer layout. Observers on a higher hierarchy layer can preempt observers on a lower hierarchy level, but not vice versa.

These statements only hold true in the case of programming complex reactive systems, in which only a subset of events is of interest in certain semantic situations. For GUIs, in which you are normally prepared to react to the majority of events the whole time, classical event handling is of course a reasonable way to go!

Take away message: Observers are more powerful than events. RAFCON goes with observers!

11.2 API

Questions that concern the core programming interface.

11.2.1 Are there examples how to use the API?

Some examples can be found in the folder `$RMPM_RAFCON_ROOT_PATH/share/examples/api` or if you use our git-repo see `$RAFCON_GIT_REPO_PATH/share/examples/api`. Many more examples of how to create a state machine using the python API can be found in `$RAFCON_GIT_REPO_PATH/source/test/common`.

11.3 GUI

Questions that concern the graphical user interface.

11.3.1 Where can I configure the RAFCON GUI?

You can either use File => Settings or manually edit `~/.config/rafcon/gui_config.yaml`. This location can also be specified by the parameter `-g` in the command line. For further explanation see [Configuration](#).

11.3.2 How to effectively collaborate on creating state machines?

The following guidelines will help you to collaboratively work on bigger state machines:

- Use git for versioning your state machines!
- Do not work at the same time on the same state machine as your colleague! You really have to know what your are doing if you merge state machine json files!
- Clearly distribute your state machine in several modules, and create library states for these modules. Then, clearly define the interfaces of these libraries. Finally, your libraries can be developed in parallel.
- If you nevertheless encounter a git conflict either throw away the smaller part of the changes, which are conflicting, and re-create them on a healthy git version. Or try to merge (recommended only for Pros!)

11.3.3 How to handle a library interface change of a library used in a (bigger) state machine ?

Even if you have a robust and clever modularization of your code, these kind of situations will occur! There are several cases.

The location of a library changed, but the library kept the same:

No problem, RAFCON will help you to relocate your libraries. Don't forget to save the library after replacing the old libraries with the new ones.

The interface of a library changed:

If you just added data ports or outcomes, you are fine! RAFCON can handle these cases easily. If you removed outcomes or data ports of a library your state machine, which includes this library can become invalid. Either data flows try to connect to no more existing data ports or transitions to no more existing outcomes. You won't be able to open the invalid state machine with the default setting. In this case use the LIBRARY_RECOVERY_MODE set to True (see Core [Configuration](#)). This will allow you to open invalid state machines. Currently, it simply removes all connections in a hierarchy if one port in the hierarchy is missing. (Removing only the erroneous connection would of course be much more convenient, and there is already an issue for that. Feel free to contribute :-)!)

The location and the interface of a library changed:

Try to avoid this case! Otherwise it will mean a good portion of work for you! The library relocation feature won't help you, as it cannot handle interface changes yet. Basically you have to cancel the library relocation process. This means that you will end up with hierarchies without connections. All the modified library states are replaced by "Hello world" dummy states. Basically, this means that you have to rebuild all hierarchies that held a link to a library, whose location and interface changed.

11.3.4 How can the hierarchy level of a state be changed in the graphical editor after it was created?

Moving a state into another state currently only works using cut and paste. As the copied state won't change its size, it is preferable to fit the sizes of the state to move and/or the target state. Then select the state to be moved and press Ctrl+X or use the menu Edit => Cut. The state is now in the clipboard, but is still shown. Now select the state into which you want to move your copied state. Make sure the target state is of type Hierarchy or Concurrency. With Ctrl+V or Edit => Paste, the original state is moved into the target state.

If you only want to combine several states you can use the group feature. This creates a new HierarchyState and moves the currently selected states into the new state. To use the group feature select all states to be grouped (they have to be on one hierarchical level) and then use the group-shortcut (STRG-G per default) or the menu bar Edit->Group entry.

11.3.5 The RAFCON GUI looks weird. Strange symbols are scattered all over the GUI. What can I do?

Probably RAFCON cannot find its fonts. If you installed RAFCON via pip, uninstall it and install it again. If you checked out RAFCON's git repo, reinstall the fonts. See the [Getting Started](#) page for that.

11.3.6 Known Issues

A window can not be un-maximized what I can do?

Generally, this is a problem related to your window manager and can be caused by different screen sizes when using several monitors or similar nasty configurations. The fastest way to solve this problem is to delete your runtime_config.yaml file which is commonly situated at `~/.config/rafcon/runtime_config.yaml` and which will be generated automatically and cleanly after removal.

Why does launching RAFCON sometimes blocks for several seconds?

This again is a problem of some window managers and is related to the automatically generated config file `~/.gtkrc`. Simply remove this file and RAFCON should start normally again.

CONTRIBUTING: DEVELOPER'S GUIDE

Everybody is encouraged to contribute to the RAFCON project. However, collaboration needs some guidelines. We try to collect all this information in this document, so please stick to this.

When contributing to this repository, please first discuss the change you wish to make via issue, email, or any other method with the owners of this repository before making a change.

Please note that *we have a code of conduct*, please follow it in all your interactions with the project.

Please check our *code style*, before making any changes to the code. Please read the *commit guidelines*, before submitting any commit.

12.1 Getting Started

For an introduction of how to install RAFCON as a user, have a look at the [website](#). Also, to get the correct dependencies, follow the instruction given on this site.

The following describes how to get the latest RAFCON version per GitHub.

First, change to the directory in which you want to clone RAFCON:

```
$ cd /some/personal/path/
```

Next, clone the [RAFCON repository](#). You can either use the HTTPS URL:

```
$ git clone https://github.com/DLR-RM/RAFCON.git
```

or the SSH URL:

```
$ git clone git@github.com:DLR-RM/RAFCON.git
```

This must of course only be done once. If you want to get the latest commits after you have cloned the repository, use

```
$ cd /some/personal/path/rafcon
$ git pull
```

In order to run RAFCON from the local code base, you have to setup the environment:

```
$ export PYTHONPATH=/some/personal/path/rafcon/source:$PYTHONPATH
$ export PATH=/some/personal/path/rafcon/bin:$PATH
```

Now you can run `rafcon` to start the RAFCON-GUI or just run `rafcon_core` to only launch the core. Hereby, `rafcon` just links to the file `/some/personal/path/rafcon/source/rafcon/gui/start.py` and `rafcon_core` points to `/some/personal/path/rafcon/source/rafcon/core/start.py`, so you could also call these files directly.

IMPORTANT: If you start rafcon for the first time start it this way:

```
$ RAFCON_CHECK_INSTALLATION=True rafcon
```

This will install all fonts and gtk-styles.

12.2 Install RAFCON from Sources

RAFCON can be also installed from our [GitHub releases](#).

If you don't want to edit the source code of RAFCON, it can be installed directly from source:

```
pip install /install/directory/rafcon/ --user
```

If you want to be able to change the source code, you can install RAFCON in editable mode.

```
pip install --editable /install/directory/rafcon/ --user
```

Any changes in /install/directory/rafcon/source will take effect when launching RAFCON.

12.3 Running and writing tests

12.3.1 Running tests with tox

The simplest and most reliable way of running the tests is using tox. If you have not installed tox, do so using

```
$ pip install tox
```

Then, in the simplest case you just call tox in the root directory of the RAFCON repository:

```
$ tox
```

This will run the following environments:

- py27, py3[4-7]: Runs the test using the according Python interpreter
- coverage: Runs the tests using Python 2.7 with a coverage report
- docs: Builds the documentation and verifies all links
- check: Verifies the sdist and wheel file

Specific environments can be run with the -e option:

```
$ tox -e 2.7,3.4  
$ tox -e docs
```

When running the tests (py27, py3[4-7] or coverage), you can pass custom options to pytest by listing them after tox [tox options] --. The default pytest options are -vx -m "(core or gui or share_elements) and not unstable".

```
$ tox -e 2.7 -- -x -m "core"  
$ tox -- -s -k "test_destruct"
```

Tox creates a virtualenv for each environment, based on the dependencies defined in `pyproject.toml` and `tox.ini`. These are only generated on the first run of an environment. If the dependencies change, you need to tell tox to recreate the environments using the `-r/--recreate` option:

```
$ tox -re py2.7
```

The RAFCON tests are annotated with a number of markers, allowing you to select specific tests:

- `core`, `gui`, `share_elements`, `network`: Tests located in a folder with that name
- `unstable`: GUI tests that do not run very reliable (e.g. because of the window manager)

Pytest allows you to select tests based on markers using the `-m` option. Markers can be combined using `not`, `and`, or `or` and parenthesis:

```
$ tox -e 2.7 -- -x -m "gui and not unstable"
```

12.3.2 Writing tests

RAFCON provides a lot of tests in the `tests/` folder. Many of these tests are integration tests, unit tests are unfortunately often missing. If a test only uses imports from `rafcon.core`, it is to be placed in `tests/core/`, otherwise in `tests/gui/`.

RAFCON uses `pytest` as testing framework. It e.g. auto detects your test files starting with `test_*`. Please have a look at the documentation before writing tests: <https://pytest.org/>

GUI tests

When you want to write an integration test using the GUI, a custom fixture named `gui` is provided (`tests/gui/conftest.py`). Simply add `gui` as parameter to your test (no import is required for tests residing beneath `test/gui/`). The fixture automatically starts the GUI before the test and closes it thereafter.

Important: Do not import any module from `rafcon.gui` outside of a function! Otherwise, models might be created withing the wrong thread, which leads to `gtkmvc` forwarding observer notifications asynchronously (via `idle_add`) instead of synchronously.

When calling an operation that causes changes (in the core, models, GUI), you need to add the operation to the GTK queue and wait until the operation has finished. This is simply done by calling `gui(function_reference, *args, **kwargs)` instead of `function_reference(*args, **kwargs)`.

If your test commands causes any `logger.warning` or `logger.error`, you need to specify the expected numbers. Do so by calling `gui.expected_warnings += 1`, respectively `gui.expected_errors += 1`, directly after the command that causes the warning/error.

The fixture will load the default core and gui config options and the libraries `generic` and `unit_test_state_machines`. If you want to override certain settings or add more libraries, use the following decorator:

```
@pytest.mark.parametrize('gui', [
    "gui_config": {
        'AUTO_BACKUP_ENABLED': True,
        'HISTORY_ENABLED': True
    },
    "libraries": {
        "ros": os.path.join(testing_utils.EXAMPLES_PATH, "libraries", "ros_libraries"),
        "turtle_libraries": os.path.join(testing_utils.EXAMPLES_PATH, "libraries",
                                         "turtle_libraries")
    }
])
```

(continues on next page)

(continued from previous page)

```

    ↵"turtle_libraries")
    }
}], indirect=True, ids=["with history, auto backup, ros and turtle libraries"])
def test_name(gui):
    pass # test code

```

Using the `ids` argument, you can specify a label for your configuration. Other possible keys are `core_config` (dict), `runtime_config` (dict) and `with_gui` (bool, for tests that operate on models but do not require the controllers and views). It is also possible to combine this with parameter sets:

```

config_options = {
    "gui_config": {
        'HISTORY_ENABLED': True
    }
}
@pytest.mark.parametrize("gui,state_path,recursive,rel_size", [
    (config_options, state_path_root, False, (40, 40)),
    (config_options, state_path_root, True, (40, 40)),
    (config_options, state_path_P, False, (20, 20)),
    (config_options, state_path_P, True, (20, 20)),
], indirect=["gui"])
def test_name(gui, state_path, recursive, rel_size, monkeypatch):
    pass # test code

```

Note that in this case, you need to set the `indirect` parameter to `["gui"]`.

The `gui` fixture offers some features:

- if you want to restart the GUI *within* a test, call `gui.restart()`
- the fixture provides shorthand access the `gui` singletons via `gui.singletons` and `core` singletons via `gui.core_singletons`, without requiring any further imports.
- if you want to run a test *after* the GUI was closed, you can set the function to be run via `gui.post_test = functools.partial(function_reference, *args, **kwargs)`

12.4 Helper Function: Convert pixels to cairo units

This function can be used when developing with cairo:

```

def pixel_to_cairo(self, pixel):
    """Helper function to convert pixels to cairo units

    The conversion is depending on the view. The critical parameter is the current
    zooming factor. The equation is:
        cairo units = pixels / zoom factor

    :param float pixel: Number of pixels to convert
    :return: Number of cairo units corresponding to given number of pixels
    :rtype: float
    """

    zoom = self.get_zoom_factor()
    return pixel / zoom

```

12.5 Code of Conduct

12.5.1 Our Pledge

In the interest of fostering an open and welcoming environment, we as contributors and maintainers pledge to making participation in our project and our community a harassment-free experience for everyone, regardless of age, body size, disability, ethnicity, gender identity and expression, level of experience, nationality, personal appearance, race, religion, or sexual identity and orientation.

12.5.2 Our Standards

Examples of behavior that contributes to creating a positive environment include:

- Using welcoming and inclusive language
- Being respectful of differing viewpoints and experiences
- Gracefully accepting constructive criticism
- Focusing on what is best for the community
- Showing empathy towards other community members

Examples of unacceptable behavior by participants include:

- The use of sexualized language or imagery and unwelcome sexual attention or advances
- Trolling, insulting/derogatory comments, and personal or political attacks
- Public or private harassment
- Publishing others' private information, such as a physical or electronic address, without explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting

12.5.3 Our Responsibilities

Project maintainers are responsible for clarifying the standards of acceptable behavior and are expected to take appropriate and fair corrective action in response to any instances of unacceptable behavior.

Project maintainers have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, or to ban temporarily or permanently any contributor for other behaviors that they deem inappropriate, threatening, offensive, or harmful.

12.5.4 Scope

This Code of Conduct applies both within project spaces and in public spaces when an individual is representing the project or its community. Examples of representing a project or community include using an official project e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event. Representation of a project may be further defined and clarified by project maintainers.

12.5.5 Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported by contacting the project team at rafcon@dlr.de. All complaints will be reviewed and investigated and will result in a response that is deemed necessary and appropriate to the circumstances. The project team is obligated to maintain confidentiality with regard to the reporter of an incident. Further details of specific enforcement policies may be posted separately.

Project maintainers who do not follow or enforce the Code of Conduct in good faith may face temporary or permanent repercussions as determined by other members of the project's leadership.

12.5.6 Attribution

This Code of Conduct is adapted from the [Contributor Covenant](https://www.contributor-covenant.org/version/1/4/), version 1.4, available at <https://www.contributor-covenant.org/version/1/4/>__

12.6 Bugs & Feature request

Please use [GitHub Issues](#) to report bugs. This page can also be used for feature requests.

12.7 Code style

12.7.1 Style guides

We follow the rules [PEP 8](#) (Style Guide for Python Code) and the DLR Coding Conventions with some minor exceptions:

- Line length is limited to 120 characters (instead of 79)
- no `__version__` in header (except `rafcon/__init__.py`)

For docstrings (and generally for the documentation), we format using `reStructuredText` for [Sphinx](#). More information on this is available in another [PEP 287](#).

12.7.2 PyCharm Coding Style/Conventions

Our recommended tool for development is PyCharm. This IDE makes it easy for you to follow our style guides. We prepared a settings file for “Code styles” and “Inspection profiles”, which you just need to import: *File > Import Settings > [project root/idea/recommended_code_style_and_inspections.jar]*.

12.8 Git commit guidelines

12.8.1 General Git commit conventions

Git is used as our version control system. Please keep the following points in mind when committing to the [repository](#):

- if you are not a core developer, all changes must be bundled in pull requests
- therefore, a new branch is required for new features and bug fixes
- try to keep pull requests small, this eases the review and will speed up the merge of the branch

- before submitting a pull request, make sure that all tests are passed
- new features require new unit tests
- each functional/logical change gets its own commit (try to keep them small)
- no excessive use of `logger.debug` outputs (in commits) and never commit `print` commands

12.8.2 Git commit messages

When looking at the Git commit history, it should be easy to see what changes have been performed. This is why it is important to have good commit messages.

What to do:

- Use imperative (`Add file ...`)
- First line is the caption of the commit (should be less than 72 chars)
 - summarizes the commit
 - mentions which code is affected (e.g. by stating the changes module/class)
- Second line is blank
- Following lines give a longer description and list changes in detail (use “-” or “*” as bullet points)
 - Why is the change necessary
 - How does it address the issue
 - Possible side effects
 - Give Issue/Feature-ID of [GitHub Issues](#)
 - * `Clos[e[s]]|Fix[e[s]]|Resolve[e[s]] #1234567` – use one of the keywords to automatically close an issue
 - * `Relates to issue #1234567` – state issue id also when the issue is not supposed to be closed

What not to do:

- Try to avoid using the `-m <msg>/--message=<msg>` flag to `git commit`
- Do not assume that the reader knows the details about bugs, features or previous commits
- Neither assume that the reader looks at the committed code

Setting up your system

Add this line to your `~/.vimrc` to add spell checking and automatic wrapping to your commit messages.

```
autocmd FileType gitcommit setlocal spell textwidth=72
```

Explaining examples

This is a good example for a commit message:

Example: Capitalized, short (<72 chars) summary

More detailed explanatory text. Wrap it to about 72 characters. Think of first line as the subject of an email and the rest of the text as the body. The blank line separating the summary from the body is critical (unless you omit the body entirely); tools like rebase can get confused if you run the two together.

(continues on next page)

(continued from previous page)

Further paragraphs come after blank lines.

- Dash '-' and asterisk '*' are both fine, followed by a space
- Use a hanging indent

The following link shows some bad examples: https://wiki.openstack.org/wiki/GitCommitMessages#Some_examples_of_bad_practice

Sources

- <https://thoughtbot.com/blog/5-useful-tips-for-a-better-commit-message>
- <https://tbaggy.com/2008/04/19/a-note-about-git-commit-messages.html>
- <https://wiki.openstack.org/wiki/GitCommitMessages>
- <https://help.github.com/en/articles/closing-issues-using-keywords>

12.9 Steps for releasing

Steps to perform, when releasing a new version of RAFCON (this section is only for releasing the tool inside our institute):

1. Decide about the new version number

Should the release be a patch or minor release? Or even a major release? Check the latest version number and adjust it appropriately.

2. Create a release Branch

Create a new branch from the latest commit you want to include into the new release. Optionally, if there are new untested feature, which you don't want to include into the release, first go back to a previous commit:

```
$ git checkout [hash]
```

Then, create the new branch and check it out:

```
$ git checkout -b release-[new version number]
```

3. Fix relevant issues (*optional*)

Within your new branch, you shouldn't integrate any new features, but only solve issues. The `develop` and `feature-xy` branches should be used for new features. Take your time to ensure that the most critical issues are fixed and RAFCON is running stable.

4. Create a test state machine for the new version (*only minor/major releases*)

For each minor release, a state machine must be created to ensure backwards compatibility using a special test.

Therefore, open the state machine in `[project directory]/tests/assets/unit_test_state_machines/backward_compatibility/[latest version number]` and save it to the same folder with the correct version as library name. Commit your changes.

5. Check tests

Run all tests and verify that they do all pass. If not, fix them! Also check the BuildBot. Commit your changes.

6. Check the changelog

Open [project directory]/doc/Changelog.rst and verify that all changes are included within the correct version number. Compare with git log and the latest closed issues on GitHub. Commit your changes.

7. Build style files

Build *.css files from *.scss files.

```
$ ./compile_scss.sh
$ git add share/themes/RAFCON/gtk-3.0/*.css --force
```

8. Apply the version number

1. If the dev dependencies have not yet been installed via pdm, then run pdm install --dev --no-editable
2. Update the version number by running pdm run bump2version [major / minor / or patch]
3. Update the date-released in [project directory]/CITATION.cff.
4. Run cffconvert --format zenodo --outfile .zenodo.json (see “[Making software citable](#)”, requires Python 3)
5. Commit and push your changes.
9. Merge to master

When everything is prepared, you can merge the release branch into the master branch:

```
$ git push release-[new version number]
$ git checkout master
$ git pull master
$ git merge release-[new version number]
$ git push
```

10. Merge to develop

Merge all changes back into the develop branch:

```
$ git checkout develop
$ git pull
$ git merge release-[new version number]
$ git push
```

11. Publish new release to PyPi

Create a new distribution file and publish it on PyPi:

```
$ rm dist/*
$ pdm build
$ twine upload dist/*
```

12. Publish to GitHub

Publish the changes to GitHub and GitHub Enterprise (assuming github is your GitHub remote name):

```
$ git push github
$ git checkout master
$ git push github
```

Make a release on GitHub by navigating to <https://github.com/DLR-RM/RAFCON/releases/new>. Enter the new version number in the “Tag version” field. Optioanlly add a release title and decription. Click “Publish release”.

13. Force build of GitHub pages

Push an empty commit to the gh-pages branch:

```
$ git checkout gh-pages
$ git commit -m 'rebuild pages' --allow-empty
$ git push
$ git push github
```

12.10 Translating RAFCON

RAFCON is prepared for internationalization (i18n) and in fact ships with a basic German translation. This translation can be extended and new translations can be added. We are happy about your support here!

12.10.1 Add translatable strings

In order to allow for translating strings in RAFCON, they need to be marked in the source code. All you need to do for this is wrapping it in the `gettext` function `_()`, e.g. `_("Hello world")`.

Strings in glade files do not need to be marked, but the `label` property needs an attribute `translatable="yes"`:

```
<object class="GtkButton">
  <property name="label" translatable="yes">Translatable button label</property>
  ...
</object>
```

12.10.2 Generating a translation template

If new translatable strings have been added or you want to create translations for a new language, a POT file (Portable Object Template) has to be created:

```
cd /dir/to/rafcon/repository
./bin/i18n-gen-msg
```

This will create a `rafcon.pot` file in the `source/rafcon/locale` folder.

12.10.3 Updating an existing translation

If you want to update a translation, open the according PO file, located in *source/rafcon/locale* (e.g. *de.po*) with Poedit. Then go to *Catalog => Update from POT file* and select the generated *rafcon.pot* file. This step is optional, if no new translatable strings have been added.

With Poedit, you can add new translations for strings comfortably. Alternatively, you can directly edit the PO files with the text editor of your choice.

12.10.4 Creating a new translation

Open Poedit. Go to *File => New From POT/PO File...* and enter the name of the new language. Start adding your translations. When you are done, store the file in *source/rafcon/locale* with the language' locale as name, e.g. *de.po* for the German translation.

12.11 HowTo on creating a splashscreen

This is a guide on creating a splash screen picture using the GIMP template.

Location The template file is located in *source/rafcon/gui/assets/themes/templates*. The standard place for images to be loaded into the startup splashscreen is *source/rafcon/gui/assets/splashscreens*.

File The template is a *.xcf* which is the native project format of GIMP. This guide is also using GIMP.

Process

1. Open the template with GIMP.
2. First select single window mode: `Windows -> Single Window Mode`. This will save you a lot of time and struggle.
3. Check if the Layer widget is present. It should contain five items, one layer Background and four RAFCON_logo layers. If not present, press `Ctrl + L`.
4. Now go to `File -> Open as Layers` and select a picture of your choice.
5. Check if the image is placed between Background and RAFCON_logo in the Layer widget. If not, drag it in the correct position.
6. Now select the layer with your picture in the Layer widget.
 - Go to the Tools widget and select the Scale tool.
 - Click on your picture with the Scale tool and fit it in the 570x320px field.
 - If your not satisfied with your result, try the “Move” tool and move the layer around.
7. Notice the small eye in the Layer widget next to a layer? Activate the visibility of a logo that serves your needs the best.
8. If everything is done, ignore the invisible layers and click on `File -> Export As`. Export the picture as whatever python is able to load into a pixelbuffer, .jpg and .png work fine.
9. Voila, there is your normed splash screen!

RAFCON API: THE RAFCON PACKAGE

RAFCON completely separates the GUI from the core, allowing state machines to be created, stored, loaded and started without having any user interface but the console.

This core of RAFCON resides in the rafcon.core sub-module. It includes classes for all types of states, a state-machine class and many more.

The rafcon.gui sub-module contains all parts needed for the GUI using GTK. It uses the Model-View-Controller-architecture (MVC).

A tutorial on how to start the core in a minimal example with a python script is given in *Starting a minimal RAFCON core (RAFCON API)*.

Finally, rafcon.utils hold several helping modules, for example for logging.

13.1 The core

Contents

- *The core*
 - *Subpackages*
 - *config*
 - *custom_exceptions*
 - *constants (in rafcon.core)*
 - *global_variable_manager*
 - *id_generator*
 - *interface*
 - *library_manager*
 - *script*
 - *singleton (in rafcon.core)*
 - *state_machine*
 - *state_machine_manager*

13.1.1 Subpackages

Execution: execution

Contents

- Execution: execution
 - base_execution_history
 - consumer_manager
 - state_machine_execution_engine
 - execution_history_factory
 - execution_history_items
 - execution_status
 - in_memory_execution_history
 - abstract_execution_history_consumer
 - file_system_consumer

base_execution_history

```
class rafcon.core.execution.base_execution_history.BaseExecutionHistory(initial_prev=None,  
                           root_state_name='',  
                           con-  
                           sumer_manager=None)
```

Bases: object

A class for the history of a state machine execution

Variables

initial_prev – optional link to a previous element for the first element pushed into this history
of type rafcon.core.execution.execution_history.HistoryItem

destroy()

Destroy the consumer manager and reset all variables

feed_consumers(execution_history_item)

Add execution history item to the dedicated queue of all consumers and notify their condition variables

Parameters

execution_history_item – the execution history item

get_last_history_item()

Returns the history item that was added last

Returns

History item added last

Return type

rafcn.core.execution.execution_history_items.HistoryItem

```
push_call_history_item(state, call_type, state_for_scoped_data, input_data=None,
link_and_feed_item_to_consumers=True)
```

Adds a new call-history-item to the history item list

A call history items stores information about the point in time where a method (entry, execute, exit) of a certain state was called.

Parameters

- **state** – the state that was called
- **call_type** – the call type of the execution step, i.e. if it refers to a container state or an execution state
- **state_for_scoped_data** – the state of which the scoped data needs to be saved for further usages (e.g. backward stepping)
- **link_and_feed_item_to_consumers** – if the history item should be feed to all other consumers

```
push_concurrency_history_item(state, number_concurrent_threads,
link_and_feed_item_to_consumers=True)
```

Adds a new concurrency-history-item to the history item list

A concurrent history item stores information about the point in time where a certain number of states is launched concurrently (e.g. in a barrier concurrency state).

Parameters

- **state** – the state that launches the state group
- **number_concurrent_threads** – the number of states that are launched
- **link_and_feed_item_to_consumers** – if the history item should be feed to all other consumers

```
push_return_history_item(state, call_type, state_for_scoped_data, output_data=None,
link_and_feed_item_to_consumers=True)
```

Adds a new return-history-item to the history item list

A return history items stores information about the point in time where a method (entry, execute, exit) of a certain state returned.

Parameters

- **state** – the state that returned
- **call_type** – the call type of the execution step, i.e. if it refers to a container state or an execution state
- **state_for_scoped_data** – the state of which the scoped data needs to be saved for further usages (e.g. backward stepping)

```
push_state_machine_start_history_item(state_machine, run_id, feed_item_to_consumers=True)
```

Adds a new state-machine-start-history-item to the history item list

A state machine starts history item stores information about the point in time where a state machine started to run

Parameters

- **state_machine** – the state machine that started
- **run_id** – the run id

- **feed_item_to_consumers** – if the history item should be feed to all other consumers

shutdown()

Destroy the consumer manager and reset all variables

consumer_manager

```
class rafcon.core.execution.consumer_manager.ExecutionHistoryConsumerManager(root_state_name)
```

Bases: `object`

A class for managing all consumers including the consumer plugins

```
FILE_SYSTEM_CONSUMER_NAME = 'file_system_consumer'
```

```
add_history_item_to_queue(execution_history_item)
```

Add execution history item to the dedicated queue of all consumers and notify their condition variables

Parameters

`execution_history_item` – the execution history item

```
property consumers_exist
```

```
property file_system_consumer_exists
```

Check if the file system consumer is activated

```
get_file_system_consumer_file_name()
```

Get the filename of the shelfe

```
register_consumer(consumer_name, consumer)
```

Register a specific consumer

Parameters

- `consumer_name` – the consumer name

- `consumer` – an instance of the consumer

```
stop_consumers()
```

Stop the working thread and unregister all consumers

```
stop_worker_thread()
```

Stop the working thread by setting interrupt to true

```
unregister_consumer(consumer)
```

Unegister a specific consumer

Parameters

`consumer` – an instance of the consumer

state_machine_execution_engine**execution_history_factory****class rafcon.core.execution.execution_history_factory.ExecutionHistoryFactory**Bases: `object`

A factory class for creating an instance of BaseExecutionHistory or InMemoryExecutionHistory

static get_execution_history(initial_prev=None, root_state_name='', consumer_manager=None)

Create an instance of a InMemoryExecutionHistory or BaseExecutionHistory

Parameters

- **initial_prev** – the initial previous history item
- **root_state_name** – the root state name
- **consumer_manager** – the consumer manager

Returns

an instance of BaseExecutionHistory or InMemoryExecutionHistory

execution_history_items**class rafcon.core.execution.execution_history_items.CallItem(state, call_type,****state_for_scoped_data, input_data,**
run_id)Bases: `ScopedDataItem`

A history item to represent a state call

to_dict(pickled=True)**rafcon.core.execution.execution_history_items.CallType**alias of `METHOD_NAME`**class rafcon.core.execution.execution_history_itemsConcurrencyItem(container_state, num-****ber_concurrent_threads,**
run_id,
consumer_manager)Bases: `HistoryItem`

A class to hold all the data for an invocation of several concurrent threads.

destroy()**to_dict(pickled=True)****class rafcon.core.execution.execution_history_itemsHistoryItem(state, run_id)**Bases: `object`

Class representing an entry within the history

An abstract class that serves as a data structure to hold all important information of a certain point in time during the execution of a state machine. A history item is an element in a doubly linked history item list.

Variables

- **state_reference** – a reference to the state performing a certain action that is going to be saved
- **path** – the state path
- **timestamp** – the time of the call/return
- **prev** – the previous history item
- **next** – the next history item

destroy()

property next

Property for the next field

property prev

Property for the prev field

property state_reference

Property for the state_reference field

to_dict(*pickled=True*)

```
class rafcon.core.execution.execution_history_items.ReturnItem(state, call_type,
                                                               state_for_scoped_data,
                                                               output_data, run_id)
```

Bases: *ScopedDataItem*

A history item to represent the return of a root state call

to_dict(*pickled=True*)

```
class rafcon.core.execution.execution_history_items.ScopedDataItem(state, call_type,
                                                               state_for_scoped_data,
                                                               child_state_input_output_data,
                                                               run_id)
```

Bases: *HistoryItem*

A abstract class to represent history items which contains the scoped data of a state

Variables

- **call_type** – the call type of the execution step, i.e. if it refers to a container state or an execution state
- **state_for_scoped_data** – the state of which the scoped data will be stored as the context data that is necessary to re-execute the state

to_dict(*pickled=True*)

```
class rafcon.core.execution.execution_history_items.StateMachineStartItem(state_machine,
                                                                           run_id)
```

Bases: *HistoryItem*

to_dict(*pickled=True*)

execution_status

```
class rafcon.core.execution.execution_status.CustomCondition(lock=None)
```

Bases: `Condition`

A class which inherits from Condition but can tell the outside world on how many threads are currently waiting.

```
get_number_of_waiting_threads()
```

A getter for the number of waiting threads :return:

```
class rafcon.core.execution.execution_status.ExecutionStatus(execution_mode=None)
```

Bases: `Observable`

A class for representing the state machine status

It inherits from Observable to make a change of its fields observable.

Variables

`execution_mode` – the execution mode of the state machine (i.e. running, paused, stopped, stepping)

```
property execution_mode
```

Property for the `_execution_mode` field

```
rafcon.core.execution.execution_status.StateMachineExecutionStatus
```

alias of `STATE_MACHINE_EXECUTION_STATUS`

in_memory_execution_history

```
class rafcon.core.execution.in_memory_execution_history.InMemoryExecutionHistory(initial_prev=None,  
root_state_name='',  
con-  
sumer_manager=None)
```

Bases: `BaseExecutionHistory`, `Observable`, `Iterable`, `Sized`

A class for the history of a state machine execution

It stores all history elements in a stack wise fashion.

Variables

`initial_prev` – optional link to a previous element for the first element pushed into this history
of type `rafcon.core.execution.execution_history.HistoryItem`

```
destroy()
```

Destroy the consumer and reset all variables

```
get_last_history_item()
```

Returns the history item that was added last

Returns

History item added last

Return type

`rafcon.core.execution.execution_history_items.HistoryItem`

```
pop_last_item(**kwargs)
```

push_call_history_item(kwargs)**

Adds a new call-history-item to the history item list

A call history items stores information about the point in time where a method (entry, execute, exit) of a certain state was called.

Parameters

- **state** – the state that was called
- **call_type** – the call type of the execution step, i.e. if it refers to a container state or an execution state
- **state_for_scoped_data** – the state of which the scoped data needs to be saved for further usages (e.g. backward stepping)
- **link_and_feed_item_to_consumers** – if the history item should be feed to all other consumers

push_concurrency_history_item(kwargs)**

Adds a new concurrency-history-item to the history item list

A concurrent history item stores information about the point in time where a certain number of states is launched concurrently (e.g. in a barrier concurrency state).

Parameters

- **state** – the state that launches the state group
- **number_concurrent_threads** – the number of states that are launched
- **link_and_feed_item_to_consumers** – if the history item should be feed to all other consumers

push_return_history_item(kwargs)**

Adds a new return-history-item to the history item list

A return history items stores information about the point in time where a method (entry, execute, exit) of a certain state returned.

Parameters

- **state** – the state that returned
- **call_type** – the call type of the execution step, i.e. if it refers to a container state or an execution state
- **state_for_scoped_data** – the state of which the scoped data needs to be saved for further usages (e.g. backward stepping)

push_state_machine_start_history_item(kwargs)**

Adds a new state-machine-start-history-item to the history item list

A state machine starts history item stores information about the point in time where a state machine started to run

Parameters

- **state_machine** – the state machine that started
- **run_id** – the run id
- **feed_item_to_consumers** – if the history item should be feed to all other consumers

shutdown()

Stop all consumers including consumer plugins

abstract_execution_history_consumer

```
class rafcon.core.execution.consumers.abstract_execution_history_consumer.  
AbstractExecutionHistoryConsumer
```

Bases: `object`

A class that should be the base for every defined consumer

consume(execution_history_item)

Override the register for the consumer to run the required procedures when a consumer wants to consume an execution history item

enqueue(execution_history_item)

Add the execution history item to the local consumer queue

Parameters

`execution_history_item` – the execution history item

register()

Override the register for the consumer to run the required procedures when a consumer starts

stop()

Stop the consumer thread

unregister()

Override the register for the consumer to run the required procedures when a consumer stops

worker()

Consume the available execution history item until the thread stops

file_system_consumer

```
class rafcon.core.execution.consumers.file_system_consumer.FileSystemConsumer(root_state_name)
```

Bases: `AbstractExecutionHistoryConsumer`

A class that consumes an execution history event and writes it onto the file system.

consume(execution_history_item)

Write to the store variable corresponding to a shelve file

register()

Open the shelve file

unregister()

Flush & close the shelve file

StateElements: state_elements**Contents**

- *StateElements: state_elements*
 - *DataFlow*
 - *DataPorts*
 - *Logical Ports*
 - *Scope (State Element)*
 - *Transition*
 - *StateElement*

DataFlow

```
class rafcon.core.state_elements.data_flow.DataFlow(from_state=None, from_key=None,
                                                    to_state=None, to_key=None,
                                                    data_flow_id=None, parent=None,
                                                    safe_init=True)
```

Bases: *StateElement*

A class for representing a data flow connection in the state machine

It inherits from Observable to make a change of its fields observable.

Variables

- **DataFlow.from_state** (*str*) – the id of the source state of the data flow connection
- **DataFlow.to_state** (*str*) – the id of the target state of the data flow connection
- **DataFlow.from_key** (*int*) – the id of the data port / scoped variable of the source state
- **DataFlow.to_key** (*int*) – the id of the data port /scoped variable of the target state
- **DataFlow.data_flow_id** (*int*) – the id of the data port, must be unique for the parent state
- **StateElement.parent** (*rafcon.core.states.container_state.ContainerState*) – reference to the parent state

property data_flow_id

Property for the _data_flow_id field

classmethod from_dict(dictionary)

Abstract method

This method must be implemented by the deriving classes. It must return an object of type cls, created from the parameters defined in dictionary. The type of cls is the type of the class the method is called on.

Parameters

dictionary (*dict*) – A Python dict with all parameters needed for creating an object of type cls

Returns
An instance of cls

Return type
cls

property from_key
Property for the _from_key field

property from_state
Property for the _from_state field

modify_origin(kwargs)**

modify_target(kwargs)**

property state_element_id
Returns the id of the state element

static state_element_to_dict(state_element)

property to_key
Property for the _to_key field

property to_state
Property for the _to_state field

DataPorts

```
class rafcon.core.state_elements.data_port.DataPort(name=None, data_type=None,
                                                    default_value=None, data_port_id=None,
                                                    parent=None, force_type=False,
                                                    init_without_default_value_type_exceptions=False,
                                                    safe_init=True)
```

Bases: *StateElement*

A class for representing a data ports in a state

Variables

- **DataPort.name** (*str*) – the name of the data port
- **DataPort.data_type** (*type*) – the value type of the data port can be handed as convertible *str* too
- **DataPort.default_value** – the default value of the data port
- **data_port_id** (*int*) – the id of the data port, must be unique for the parent state
- **StateElement.parent** (*rafcon.core.states.state.State*) – reference to the parent state
- **DataPort.force_type** (*bool*) – if true the DataPort type exception is not raised while initiation (backward compatibility)
- **DataPort.init_without_default_value_type_exceptions** (*bool*) – if true it is allowed to initiate with any default value type used to load not matching default value data types and correct them using the GUI.

change_data_type(**kwargs)

check_default_value(default_value, data_type=None)

Check whether the passed default value suits to the passed data type. If no data type is passed, the data type of the data port is used. If the default value does not fit, an exception is thrown. If the default value is of type string, it is tried to convert that value to the data type.

Parameters

- **default_value** – The default value to check
- **data_type** – The data type to use

Raises

exceptions.AttributeError – if check fails

Returns

The converted default value

property data_port_id

Property for the _data_port_id field

property data_type

Property for the _data_type field

property default_value

Property for the _default_value field

classmethod from_dict(dictionary)

Abstract method

This method must be implemented by the deriving classes. It must return an object of type cls, created from the parameters defined in dictionary. The type of cls is the type of the class the method is called on.

Parameters

dictionary (*dict*) – A Python dict with all parameters needed for creating an object of type
cls

Returns

An instance of cls

Return type

cls

property name

Property for the _name field

property state_element_id

Returns the id of the state element

static state_element_to_dict(state_element)

class rafcon.core.state_elements.data_port.InputDataPort(name=None, data_type=None,
default_value=None, data_port_id=None,
parent=None, force_type=False,
init_without_default_value_type_exceptions=False,
safe_init=True)

Bases: *DataPort*

```
class rafcon.core.state_elements.data_port.OutputDataPort(name=None, data_type=None,
                                                       default_value=None,
                                                       data_port_id=None, parent=None,
                                                       force_type=False,
                                                       init_without_default_value_type_exceptions=False,
                                                       safe_init=True)
```

Bases: *DataPort*

Logical Ports

```
class rafcon.core.state_elements.logical_port.Income(parent=None, safe_init=True)
```

Bases: *LogicalPort*

A class for representing an income of a state

There can only be one Income within a state, which is why no income_id is required.

Variables

StateElement.parent (*rafcon.core.states.state.State*) – reference to the parent state

classmethod from_dict(dictionary)

Abstract method

This method must be implemented by the deriving classes. It must return an object of type *cls*, created from the parameters defined in *dictionary*. The type of *cls* is the type of the class the method is called on.

Parameters

dictionary (*dict*) – A Python dict with all parameters needed for creating an object of type *cls*

Returns

An instance of *cls*

Return type

cls

property state_element_id

Always returns 0

Only required for consistent handling of all state elements.

Returns

0

static state_element_to_dict(state_element)

```
class rafcon.core.state_elements.logical_port.LogicalPort(parent=None, safe_init=True)
```

Bases: *StateElement*

Base class for the logical ports

```
class rafcon.core.state_elements.logical_port.Outcome(outcome_id=None, name=None,
                                                       parent=None, safe_init=True)
```

Bases: *LogicalPort*

A class for representing an outcome of a state

As the name of an outcome can be changes without modifying the transitions the primary key of an outcome is its id and not its name.

Variables

- **Outcome.outcome_id** (*int*) – the id of the outcome, must be unique on one hierarchy level
- **Outcome.name** (*str*) – the human readable name of the outcome
- **StateElement.parent** (`rafcon.core.states.state.State`) – reference to the parent state

classmethod from_dict(dictionary)

Abstract method

This method must be implemented by the deriving classes. It must return an object of type `cls`, created from the parameters defined in `dictionary`. The type of `cls` is the type of the class the method is called on.

Parameters

dictionary (*dict*) – A Python dict with all parameters needed for creating an object of type `cls`

Returns

An instance of `cls`

Return type

`cls`

property name

Returns the Outcome's name

property outcome_id

Returns the outcome_id

property state_element_id

Returns the id of the state element

static state_element_to_dict(state_element)

Scope (State Element)

class rafcon.core.state_elements.scope.ScopedData(name, value, value_type, from_state, data_port_type, parent=None, safe_init=True)

Bases: `StateElement`

A class for representing scoped data of a container state

It inherits from `Observable` to make a change of its fields observable.

Variables

- **ScopedData.name** (*str*) – the name of the scoped data
- **ScopedData.from_state** (*str*) – the state_id of the state that wrote to the scoped data last
- **value_type** (*type*) – specifies the type of self._value; the setter of __value will only allow assignments that satisfies the data_type constraint
- **value** – the current value of the scoped data
- **data_port_type** – the type of the data port that wrote to the scoped data last
- **timestamp** (*str*) – the timestamp when the scoped data was written to last

property data_port_type

Property for the _data_port_type field

classmethod from_dict(dictionary)

Abstract method

This method must be implemented by the deriving classes. It must return an object of type cls, created from the parameters defined in dictionary. The type of cls is the type of the class the method is called on.

Parameters

dictionary (*dict*) – A Python dict with all parameters needed for creating an object of type cls

Returns

An instance of cls

Return type

cls

property from_state

Property for the _from_state field

property name

Property for the _name field

property state_element_id

Returns the id of the state element

static state_element_to_dict(state_element)**property timestamp**

Property for the _timestamp field

property value

Property for the _value field

property value_type

Property for the _value_type field

```
class rafcon.core.state_elements.scope.ScopedVariable(name=None, data_type=None,
                                                    default_value=None,
                                                    scoped_variable_id=None, parent=None,
                                                    safe_init=True)
```

Bases: *DataPort*

A class for representing a scoped variable in a container state

It inherits from the DataPort class as it needs exactly the same class fields. It inherits from Observable to make a change of its fields observable.

Variables

- **DataPort.name** (*str*) – the name of the scoped variable
- **DataPort.data_type** (*type*) – specifies the type of the scoped variable (data port); the setter of _value will only allow assignments that satisfies the type constraint
- **DataPort.default_value** – specifies the default value of the scoped variable (data port)
- **scoped_variable_id** (*int*) – the id of the scoped variable (see DataPort.data_port_id), must be unique for the parent state

- **StateElement.parent** (rafcon.core.states.container_state.ContainerState) – reference to the parent state

classmethod from_dict(dictionary)

Abstract method

This method must be implemented by the deriving classes. It must return an object of type cls, created from the parameters defined in dictionary. The type of cls is the type of the class the method is called on.

Parameters

dictionary (dict) – A Python dict with all parameters needed for creating an object of type cls

Returns

An instance of cls

Return type

cls

static state_element_to_dict(state_element)

rafcon.core.state_elements.scope.generate_time_stamp()

Generate a time stamp for the current time as integer in micro. :return:

Transition

class rafcon.core.state_elements.transition.Transition(from_state, from_outcome, to_state, to_outcome, transition_id, parent=None, safe_init=True)

Bases: *StateElement*

A class for representing a transition in the state machine

It inherits from Observable to make a change of its fields observable.

Raises an exceptions.TypeError if the transition_id is not of type int.

Variables

- **transition_id (int)** – the id of the transition, must be unique for the parent state
- **Transition.from_state (str)** – the source state of the transition
- **Transition.from_outcome (int)** – the outcome of the source state
- **Transition.to_state (str)** – the target state of the transition
- **Transition.to_outcome (int)** – the outcome of the target state
- **StateElement.parent** (rafcon.core.states.container_state.ContainerState) – reference to the parent state

classmethod from_dict(dictionary)

Abstract method

This method must be implemented by the deriving classes. It must return an object of type cls, created from the parameters defined in dictionary. The type of cls is the type of the class the method is called on.

Parameters

dictionary (dict) – A Python dict with all parameters needed for creating an object of type cls

Returns
An instance of cls

Return type
cls

property from_outcome
Property for the _from_outcome field

property from_state
Property for the _from_state field

modify_origin(kwargs)**

modify_target(kwargs)**

property state_element_id
Returns the id of the state element

static state_element_to_dict(state_element)

property to_outcome
Property for the to_outcome field

property to_state
Property for the _to_state field

property transition_id
Property for the _transition_id field

StateElement

class rafcon.core.state_elements.state_element.StateElement(parent=None, safe_init=True)

Bases: Observable, YAMLObject, JSONObject, Hashable

A abstract base class for all elements of a state (ports, connections)

It inherits from Observable to make a change of its fields observable. It also inherits from YAMLObject, in order to store/load it as YAML file.

It raises an exceptions.NotImplementedError if the type of the class instance is type(self).

Variables

StateElement.parent ([rafcon.core.states.state.State](#)) – Parent state of the state element

property core_element_id

Returns the id of the state element

classmethod from_dict(dictionary)

Abstract method

This method must be implemented by the deriving classes. It must return an object of type cls, created from the parameters defined in dictionary. The type of cls is the type of the class the method is called on.

Parameters

dictionary ([dict](#)) – A Python dict with all parameters needed for creating an object of type cls

Returns

An instance of `cls`

Return type

`cls`

property parent

Getter for the parent state of the state element

Returns

None if parent is not defined, else the parent state

Return type

`rafcon.core.states.state.State`

property state_element_id

Returns the id of the state element

static state_element_to_dict(state_element)**to_dict()**

Abstract method

This method must be implemented by the deriving classes. It must return a Python dict with all parameters of self, which are needed to create a copy of self.

Returns

A Python dict with all needed parameters of self

Return type

`dict`

update_hash(obj_hash)

Should be implemented by derived classes to update the hash with their data fields

Parameters

`obj_hash` – The hash object (see Python hashlib)

States: states**Contents**

- *States: states*
 - *barrier_concurrency*
 - *concurrency_state*
 - *container_state*
 - *execution_state*
 - *hierarchy_state*
 - *library_state*
 - *preemptive_concurrency*
 - *state*

barrier_concurrency

```
class rafcon.core.states.barrier_concurrency_state.BarrierConcurrencyState(name=None,
                                                               state_id=None, in-
                                                               put_data_ports=None,
                                                               out-
                                                               put_data_ports=None,
                                                               income=None,
                                                               outcomes=None,
                                                               states=None,
                                                               transitions=None,
                                                               data_flows=None,
                                                               start_state_id=None,
                                                               scoped_variables=None,
                                                               de-
                                                               cider_state=None,
                                                               load_from_storage=False,
                                                               safe_init=True)
```

Bases: *ConcurrencyState*

The barrier concurrency holds a list of states that are executed in parallel. It waits until all states finished their execution before it returns.

Note: In the backward execution case the decider state does not have to be backward executed, as it only decides the outcome of the barrier concurrency state. In a backward execution the logic flow obviously already exists.

The order of history items for the concurrency state is: Call - Concurrency - Return and for the backward case: Return - Concurrency - Call

For the children of the concurrency state the history items are: In the forward case: - Call: Before calling the child - Return: After executing the child In the backward case: - Pop Return: Before backward executing the child - Pop Call: After backward executing the child

The decider state is not considered in the backward execution case.

add_state(state, storage_load=False)

Overwrite the parent class add_state method

Add automatic transition generation for the decider_state.

Parameters

state – The state to be added

Returns

classmethod from_dict(dictionary)

An abstract method each state has to implement.

Parameters

dictionary – the dictionary of parameters a state can be created from

Raises

exceptions.NotImplementedError – in any case

remove_state(state_id, recursive=True, force=False, destroy=True)

Overwrite the parent class remove state method by checking if the user tries to delete the decider state

Parameters

- **state_id** – the id of the state to remove
- **recursive** – a flag to indicate a recursive disassembling of all substates
- **force** – a flag to indicate forcefully deletion of all states (important of the decider state in the barrier concurrency state)
- **destroy** – a flag which indicates if the state should not only be disconnected from the state but also destroyed, including all its state elements

Raises

`exceptions.AttributeError` – if the state_id parameter is the decider state

run()

This defines the sequence of actions that are taken when the barrier concurrency state is executed

Returns**run_decider_state(decider_state, child_errors, final_outcomes_dict)**

Runs the decider state of the barrier concurrency state. The decider state decides on which outcome the barrier concurrency is left.

Parameters

- **decider_state** – the decider state of the barrier concurrency state
- **child_errors** – error of the concurrent branches
- **final_outcomes_dict** – dictionary of all outcomes of the concurrent branches

Returns**property states**

Property for the `_states` field

The setter-method substitute `ContainerState.states` which is a dict. The method checks if the elements are of the right type or will cancel the operation and recover old outcomes. The method does check validity of the elements by calling the parent-setter.

```
class rafcon.core.states.barrier_concurrency_state.DeciderState(name=None, state_id=None,
                                                               input_data_ports=None,
                                                               output_data_ports=None,
                                                               income=None, outcomes=None,
                                                               path=None, filename=None,
                                                               safe_init=True)
```

Bases: `ExecutionState`

A class to represent a state for deciding the exit of barrier concurrency state.

This type of `ExecutionState` has initial always the `UNIQUE_DECIDER_STATE_ID`.

get_errors_for_state_name(name)

Returns the error message of the child state specified by name.

Note: This is utility function that is used by the programmer to make a decision based on the final outcome of its child states. A state is not uniquely specified by the name, but as the programmer normally does not want to use state-ids in his code this utility function was defined.

Parameters

name – The name of the state to get the error message for

Returns

get_outcome_for_state_id(state_id)

Returns the final outcome of the child state specified by the state_id.

Parameters

state_id – The id of the state to get the final outcome for.

Returns**get_outcome_for_state_name(name)**

Returns the final outcome of the child state specified by name.

Note: This is utility function that is used by the programmer to make a decision based on the final outcome of its child states. A state is not uniquely specified by the name, but as the programmer normally does not want to use state-ids in his code this utility function was defined.

Parameters

name – The name of the state to get the final outcome for.

Returns**concurrency_state**

```
class rafcon.core.states.concurrency_state.ConcurrencyState(name=None, state_id=None,
                                                               input_keys=None, output_keys=None,
                                                               income=None, outcomes=None,
                                                               states=None, transitions=None,
                                                               data_flows=None,
                                                               start_state_id=None,
                                                               scoped_variables=None,
                                                               safe_init=True)
```

Bases: *ContainerState*

A class to represent a concurrency state for the state machine

The concurrency state holds several child states, that can be container states again

finalize_backward_execution()

Utility function to finalize the backward execution of the concurrency state.

Returns**finalize_concurrency_state(outcome)**

Utility function to finalize the forward execution of the concurrency state.

Parameters

outcome –

Returns**join_state(state, history_index, concurrency_history_item)**

a utility function to join a state

Parameters

- **state** – the state to join
- **history_index** – the index of the execution history stack in the concurrency history item for the given state
- **concurrency_history_item** – the concurrency history item that stores the execution history stacks of all children

Returns**run(*args, **kwargs)**

The abstract run method that has to be implemented by all concurrency states.

Parameters

- **args** – list of optional arguments
- **kwargs** – optional keyword arguments

Raises

exceptions.AttributeError – if this method was not implemented by the subclass

setup_forward_or_backward_execution()

Sets up the execution of the concurrency states dependent on if the state is executed forward or backward.

Returns**start_child_states(concurrency_history_item, do_not_start_state=None)**

Utility function to start all child states of the concurrency state.

Parameters

- **concurrency_history_item** – each concurrent child branch gets an execution history stack of this concurrency history item
- **do_not_start_state** – optionally the id of a state can be passed, that must not be started (e.g. in the case of the barrier concurrency state the decider state)

Returns**container_state**

```
class rafcon.core.states.container_state.ContainerState(name=None, state_id=None,
                                                       input_data_ports=None,
                                                       output_data_ports=None, income=None,
                                                       outcomes=None, states=None,
                                                       transitions=None, data_flows=None,
                                                       start_state_id=None,
                                                       scoped_variables=None,
                                                       missing_library_meta_data=None,
                                                       is_dummy=False, safe_init=True)
```

Bases: *State*

A class for representing a state in the state machine

Only the variables are listed that are not already contained in the state base class

Variables

- **ContainerState.states** (*dict*) – the child states of the container state of the state
- **ContainerState.transitions** (*dict*) – transitions between all child states
- **ContainerState.data_flows** (*dict*) – data flows between all child states
- **ContainerState.start_state_id** (*str*) – the state to start with when the hierarchy state is executed
- **ContainerState.scoped_variables** (*dict*) – the scoped variables of the container

add_data_flow(kwargs)**

add_default_values_of_scoped_variables_to_scoped_data()

Add the scoped variables default values to the scoped_data dictionary

add_input_data_to_scoped_data(dictionary)

Add a dictionary to the scoped data

As the input_data dictionary maps names to values, the functions looks for the proper data_ports keys in the input_data_ports dictionary

Parameters

- **dictionary** – The dictionary that is added to the scoped data
- **state** – The state to which the input_data was passed (should be self in most cases)

add_scoped_variable(kwargs)**

add_state(kwargs)**

add_state_execution_output_to_scoped_data(dictionary, state)

Add a state execution output to the scoped data

Parameters

- **dictionary** – The dictionary that is added to the scoped data
- **state** – The state that finished execution and provide the dictionary

add_transition(kwargs)**

change_state_id(state_id=None)

Changes the id of the state to a new id. This functions replaces the old state_id with the new state_id in all data flows and transitions.

Parameters

state_id – The new state if of the state

change_state_type(kwargs)**

check_child_validity(child)

Check validity of passed child object

The method is called by state child objects (transitions, data flows) when these are initialized or changed. The method checks the type of the child and then checks its validity in the context of the state.

Parameters

child (object) – The child of the state that is to be tested

Return bool validity, str message

validity is True, when the child is valid, False else. message gives more information especially if the child is not valid

check_data_flow_id(data_flow_id)

Check the data flow id and calculate a new one if its None

Parameters

data_flow_id – The data flow id to check

Returns

The new data flow id

Raises

exceptions.AttributeError – if data_flow.data_flow_id already exists

check_data_port_connection(check_data_port)

Checks the connection validity of a data port

The method is called by a child state to check the validity of a data port in case it is connected with data flows. The data port does not belong to ‘self’, but to one of self.states. If the data port is connected to a data flow, the method checks, whether these connect consistent data types of ports.

Parameters

check_data_port (`rafcon.core.data_port.DataPort`) – The port to check

Returns

valid, message

check_transition_id(transition_id)

Check the transition id and calculate a new one if its None

Parameters

transition_id – The transition-id to check

Returns

The new transition id

Raises

exceptions.AttributeError – if transition.transition_id already exists

property data_flows

Property for the _data_flows field

The setter-method substitute ContainerState._data_flows with handed dictionary. The method checks if the elements are of the right type and the keys consistent (DataFlow.data_flow_id==key). The method does check validity of the elements by calling the parent-setter and in case of failure cancel the operation and recover old _data_flows dictionary.

Returns

Dictionary data_flows[data_flow_id] of `rafcon.core.data_flow.DataFlow`

Return type

dict

destroy(recursive)

Removes all the state elements.

Parameters

recursive – Flag whether to destroy all state elements which are removed

classmethod from_dict(dictionary)

An abstract method each state has to implement.

Parameters

dictionary – the dictionary of parameters a state can be created from

Raises

exceptions.NotImplementedError – in any case

get_connections_for_state(state_id)

The method generates two dictionaries with related transitions and data flows for the given state_id

The method creates dictionaries for all ‘internal’ and ‘external’ (first dict-key) connections of the state. Both dictionaries contain sub dicts with 3 (external)/4 (internal) fields ‘enclosed’, ‘ingoing’, ‘outgoing’

and ‘self’. - ‘enclosed’ means the handed state.states cover origin and target of those linkage - ‘ingoing’ means the handed state is target of those linkage - ‘outgoing’ means the handed state is origin of those linkage - ‘self’ (corner case) single state that has linkage with it self and is thereby also origin and target at the same time

Parameters

state_id – State taken into account.

Return type

tuple

Returns

related_transitions, related_data_flows

`get_connections_for_state_and_scoped_variables(state_ids, scoped_variables)`

The method generates two dictionaries with transitions and data flows for the given state ids and scoped vars

The method creates dictionaries with connections for a set of states and scoped variables. Both dictionaries have 3 fields (as first dict-key), ‘enclosed’, ‘ingoing’ and ‘outgoing’ - ‘enclosed’ means the given sets cover origin and target of those linkage - ‘ingoing’ means the given sets is target of those linkage - ‘ingoing’ means the given sets is origin of those linkage

Parameters

- **state_ids** – List of states taken into account.
- **scoped_variables** – List of scoped variables taken into account

Return type

tuple

Returns

related_transitions, related_data_flows

`get_data_port(state_id, port_id)`

Searches for a data port

The data port specified by the state id and data port id is searched in the state itself and in its children.

Parameters

- **state_id** (*str*) – The id of the state the port is in
- **port_id** (*int*) – The id of the port

Returns

The searched port or None if it is not found

`get_data_port_by_id(data_port_id)`

Search for the given data port id in the data ports of the state

The method tries to find a data port in the input and output data ports as well as in the scoped variables.

Parameters

data_port_id – the unique id of the data port

Returns

the data port with the searched id or None if not found

`get_data_port_ids()`

get_inputs_for_state(state)

Retrieves all input data of a state. If several data flows are connected to an input port the most current data is used for the specific input port.

Parameters

state – the state of which the input data is determined

Returns

the input data of the target state

get_number_of_data_flows()

Returns the number of data flows :return:

get_number_of_transitions()

Returns the number of transitions :return:

get_outcome(state_id, outcome_id)**get_scoped_variable_from_name(name)**

Get the scoped variable for a unique name

Parameters

name – the unique name of the scoped variable

Returns

the scoped variable specified by the name

Raises

exceptions.AttributeError – if the name is not in the the scoped_variables dictionary

get_start_state(set_final_outcome=False)

Get the start state of the container state

Parameters

set_final_outcome – if the final_outcome of the state should be set if the income directly connects to an outcome

Returns

the start state

get_state_for_transition(transition)

Calculate the target state of a transition

Parameters

transition – The transition of which the target state is determined

Returns

the to-state of the transition

Raises

exceptions.TypeError – if the transition parameter is of wrong type

get_states_statistics(hierarchy_level)

Returns the numer of child states :return:

get_transition_for_outcome(state, outcome)

Determines the next transition of a state.

Parameters

- **state** – The state for which the transition is determined

- **outcome** – The outcome of the state, that is given in the first parameter

Returns

the transition specified by the the state and the outcome

Raises

`exceptions.TypeError` – if the types of the passed parameters are incorrect

group_states(kwargs)**

handle_no_start_state()

Handles the situation, when no start state exists during execution

The method waits, until a transition is created. It then checks again for an existing start state and waits again, if this is not the case. It returns the None state if the the state machine was stopped.

handle_no_transition(state)

This function handles the case that there is no transition for a specific outcome of a sub-state.

The method waits on a condition variable to a new transition that will be connected by the programmer or GUI-user.

Parameters

state – The sub-state to find a transition for

Returns

The transition for the target state.

Raises

`exceptions.RuntimeError` – if the execution engine is stopped (this will be caught at the end of the run method)

property is_dummy

Property for the _is_dummy field

property missing_library_meta_data

Property for the _missing_library_meta_data field

recursively_pause_states()

Pause the state and all of it child states.

recursively_preempt_states()

Preempt the state and all of it child states.

recursively_resume_states()

Resume the state and all of it child states.

remove(state_element, recursive=True, force=False, destroy=True)

Remove item from state

Parameters

- **state_element** (`StateElement`) – State or state element to be removed
- **recursive** (`bool`) – Only applies to removal of state and decides whether the removal should be called recursively on all child states
- **force** (`bool`) – if the removal should be forced without checking constraints
- **destroy** (`bool`) – a flag that signals that the state element will be fully removed and disassembled

remove_data_flow(kwargs)**

remove_data_flows_with_data_port_id(data_port_id)

Remove an data ports whose from_key or to_key equals the passed data_port_id

Parameters

data_port_id (`int`) – the id of a data_port of which all data_flows should be removed, the id can be a input or output data port id

remove_outcome_hook(outcome_id)

Removes internal transition going to the outcome

remove_scoped_variable(kwargs)**

remove_state(kwargs)**

remove_transition(kwargs)**

run(*args, **kwargs)

Implementation of the abstract run() method of the `threading.Thread`

Should be filled with code, that should be executed for each container_state derivative. :raises exceptions.NotImplementedError: in every case

property scoped_data

Property for the _scoped_data field

property scoped_variables

Property for the _scoped_variables field

The setter-method ContainerState._scoped_variables with a handed dictionary. The method checks if the elements are of the right type and the keys consistent (Transition.transition_id==key). The method does check validity of the elements by calling the parent-setter and in case of failure cancel the operation and recover old _scoped_variables dictionary.

Returns

Dictionary scoped_variables[data_port_id] of `rafcon.core.scope.ScopedVariable`

Return type

`dict`

set_start_state(state)

Sets the start state of a container state

Parameters

state – The state_id of a state or a direct reference of the state (that was already added to the container) that will be the start state of this container state.

setup_run()

Executes a generic set of actions that has to be called in the run methods of each derived state class.

Returns

property start_state_id

The start state is the state to which the first transition goes to.

The setter-method creates a unique first transition to the state with the given id. Existing first transitions are removed. If the given state id is None, the first transition is removed.

Returns

The id of the start state

static state_to_dict(state)

property states

Property for the _states field

The setter-method substitute ContainerState.states which is a dict. The method checks if the elements are of the right type or will cancel the operation and recover old outcomes. The method does check validity of the elements by calling the parent-setter.

substitute_state(kwargs)**

property transitions

Property for the _transitions field

The setter-method substitute ContainerState._transitions with a handed dictionary. The method checks if the elements are of the right type and the keys consistent (Transition.transition_id==key). The method does check validity of the elements by calling the parent-setter and in case of failure cancel the operation and recover old _transitions dictionary.

Returns

Dictionary transitions[transition_id] of rafcon.core.transition.Transition

Return type

dict

ungroup_state(kwargs)**

update_hash(obj_hash)

Should be implemented by derived classes to update the hash with their data fields

Parameters

obj_hash – The hash object (see Python hashlib)

update_scoped_variables_with_output_dictionary(dictionary, state)

Update the values of the scoped variables with the output dictionary of a specific state.

Param

the dictionary to update the scoped variables with

Param

the state the output dictionary belongs to

write_output_data(specific_output_dictionary=None)

Write the scoped data to the output of the state. Called before exiting the container state.

Parameters

specific_output_dictionary – an optional dictionary to write the output data in

Returns

execution_state

```
class rafcon.core.states.execution_state.ExecutionState(name=None, state_id=None,
                                                       input_data_ports=None,
                                                       output_data_ports=None, income=None,
                                                       outcomes=None, path=None,
                                                       filename=None, check_path=True,
                                                       safe_init=True)
```

Bases: `State`

A class to represent a state for executing arbitrary functions

This kind of state does not have any child states.

classmethod from_dict(dictionary)

An abstract method each state has to implement.

Parameters

`dictionary` – the dictionary of parameters a state can be created from

Raises

`exceptions.NotImplementedError` – in any case

property name

Property for the `_name` field

Return type

`str`

Returns

Name of state

run()

This defines the sequence of actions that are taken when the execution state is executed

Returns

property script

Returns the property for the `_script` field.

property script_text

update_hash(obj_hash)

Should be implemented by derived classes to update the hash with their data fields

Parameters

`obj_hash` – The hash object (see Python hashlib)

hierarchy_state

```
class rafcon.core.states.hierarchy_state.HierarchyState(name=None, state_id=None,
                                                       input_data_ports=None,
                                                       output_data_ports=None, income=None,
                                                       outcomes=None, states=None,
                                                       transitions=None, data_flows=None,
                                                       start_state_id=None,
                                                       scoped_variables=None,
                                                       missing_library_meta_data=None,
                                                       is_dummy=False, safe_init=True)
```

Bases: `ContainerState`

A class to represent a hierarchy state for the state machine

The hierarchy state holds several child states, that can be container states on their own

`check_if_child_state_was_modified()`

`run()`

This defines the sequence of actions that are taken when the hierarchy is executed. A hierarchy state executes all its child states recursively. Principally this code collects all input data for the next child state, executes it, stores its output data and determines the next state based on the outcome of the child state. :return:

`library_state`

```
class rafcon.core.states.library_state.LibraryState(library_path=None, library_name=None,
                                                    version=None, name=None, state_id=None,
                                                    income=None, outcomes=None,
                                                    input_data_port_runtime_values=None,
                                                    use_runtime_value_input_data_ports=None,
                                                    output_data_port_runtime_values=None,
                                                    use_runtime_value_output_data_ports=None,
                                                    allow_user_interaction=True, safe_init=True,
                                                    skip_runtime_data_initialization=False)
```

Bases: `State`

A class to represent a library state for the state machine

Only the variables are listed that are not already contained in the state base class.

The constructor uses an exceptions.AttributeError if the passed version of the library and the version found in the library paths do not match.

Variables

- `library_path` (`str`) – the path of the library relative to a certain library path (e.g. lwr/gripper/)
- `library_name` (`str`) – the name of the library between all child states: (e.g. open, or close)
- `State.name` (`str`) – the name of the library state
- `State.state_id` (`str`) – the id of the library state
- `input_data_port_runtime_values` (`dict`) – a dict to store all the runtime values for the input data ports
- `use_runtime_value_input_data_ports` (`dict`) – flags to indicate if the runtime or the default value should be used for a specific input data port
- `output_data_port_runtime_values` (`dict`) – a dict to store all the runtime values for the output data ports
- `use_runtime_value_output_data_ports` (`dict`) – flags to indicate if the runtime or the default value should be used for a specific output data port
- `allow_user_interaction` (`dict`) – flag to indicate if the user can support in localizing moved libraries

- **skip_runtime_data_initialization** – flag to indicate if the runtime-data data structures have to be initialized, this is not needed e.g. in the case of a copy

add_input_data_port(*name*, *data_type*=None, *default_value*=None, *data_port_id*=None)

Overwrites the add_input_data_port method of the State class. Prevents user from adding a output data port to the library state.

For further documentation, look at the State class. :raises exceptions.NotImplementedError: in any case

add_outcome(*name*, *outcome_id*=None)

Overwrites the add_outcome method of the State class. Prevents user from adding a outcome to the library state.

For further documentation, look at the State class.

Raises

exceptions.NotImplementedError – in any case

add_output_data_port(*name*, *data_type*, *default_value*=None, *data_port_id*=None)

Overwrites the add_output_data_port method of the State class. Prevents user from adding a output data port to the library state.

For further documentation, look at the State class. :raises exceptions.NotImplementedError: in any case

destroy(*recursive*=True)

Removes all the state elements.

Parameters

recursive – Flag wether to destroy all state elements which are removed

classmethod from_dict(*dictionary*)

An abstract method each state has to implement.

Parameters

dictionary – the dictionary of parameters a state can be created from

Raises

exceptions.NotImplementedError – in any case

get_number_of_data_flows()

Return the number of data flows for a state. Per default states do not have data flows. :return:

get_number_of_transitions()

Return the number of transitions for a state. Per default states do not have transitions. :return:

get_states_statistics(*hierarchy_level*)

Returns the numer of child states. As per default states do not have child states return 1. :return:

get_storage_path(*appendix*=None)

Recursively create the storage path of the state.

The path is generated in bottom up method i.e. from the nested child states to the root state. The method concatenates the concatenation of (State.name and State.state_id) as state identifier for the path.

Parameters

appendix (*str*) – the part of the path that was already calculated by previous function calls

Return type

str

Returns

the full path to the root state

property input_data_port_runtime_values

Property for the _input_data_port_runtime_values field

property library_hierarchy_depth

Calculates the library hierarchy depth

Counting starts at the current library state. So if there is no upper library state the depth is one.

Returns

library hierarchy depth

Return type

`int`

property library_name

Property for the _library_name field

property library_path

Property for the _library_path field

property output_data_port_runtime_values

Property for the _output_data_port_runtime_values field

recursively_pause_states()

Pause the state and all of its child states.

recursively_preempt_states()

Preempt the state and all of its child states.

recursively_resume_states()

Resume the state and all of its child states.

remove_input_data_port(`data_port_id`, `force=False`, `destroy=True`)

Overwrites the remove_input_data_port method of the State class. Prevents user from removing an input data port from the library state.

For further documentation, look at the State class.

Parameters

`force (bool)` – True if the removal should be forced

Raises

`exceptions.NotImplementedError` – in the removal is not forced

remove_outcome(`outcome_id`, `force=False`, `destroy=True`)

Overwrites the remove_outcome method of the State class. Prevents user from removing an outcome from the library state.

For further documentation, look at the State class.

Raises

`exceptions.NotImplementedError` – in any case

remove_output_data_port(`data_port_id`, `force=False`, `destroy=True`)

Overwrites the remove_output_data_port method of the State class. Prevents user from removing an output data port from the library state.

For further documentation, look at the State class. :param bool force: True if the removal should be forced
:raises exceptions.NotImplementedError: in the removal is not forced

run()

This defines the sequence of actions that are taken when the library state is executed

It basically just calls the run method of the container state :return:

```
set_input_runtime_value(**kwargs)
set_output_runtime_value(**kwargs)
set_use_input_runtime_value(**kwargs)
set_use_output_runtime_value(**kwargs)
```

property state_copy

Property for the _state_copy field

```
static state_to_dict(state)
```

```
update_hash(obj_hash)
```

Should be implemented by derived classes to update the hash with their data fields

Parameters

obj_hash – The hash object (see Python hashlib)

```
property use_runtime_value_input_data_ports
```

Property for the _use_runtime_value_input_data_ports field

```
property use_runtime_value_output_data_ports
```

Property for the _use_runtime_value_output_data_ports field

```
property version
```

Property for the _version field

preemptive_concurrency

```
class rafcon.core.states.preemptive_concurrency_state.PreemptiveConcurrencyState(name=None,
                                     state_id=None,
                                     in-
                                     put_data_ports=None,
                                     out-
                                     put_data_ports=None,
                                     in-
                                     come=None,
                                     out-
                                     comes=None,
                                     states=None,
                                     transi-
                                     tions=None,
                                     data_flows=None,
                                     start_state_id=None,
                                     scoped_variables=None,
                                     safe_init=True)
```

Bases: *ConcurrencyState*

The preemptive concurrency state has a set of substates which are started when the preemptive concurrency state executes. The execution of preemptive concurrency state waits for the first substate to return, preempts all other substates and finally returns self.

run()

This defines the sequence of actions that are taken when the preemptive concurrency state is executed

Returns**state**

```
class rafcon.core.states.state.State(name=None, state_id=None, input_data_ports=None,
                                     output_data_ports=None, income=None, outcomes=None,
                                     parent=None, safe_init=True)
```

Bases: Observable, YAMLObject, JSONObject, Hashable

A class for representing a state in the state machine

It inherits from Observable to make a change of its fields observable.

Variables

- **State.name** (*str*) – the name of the state
- **State.state_id** (*str*) – the id of the state
- **State.input_data_ports** (*dict*) – holds the input data ports of the state
- **State.output_data_ports** (*dict*) – holds the output data ports of the state
- **State.outcomes** (*dict*) – holds the state outcomes, which are the connection points for transitions
- **State.parent** (*rafcon.core.states.container_state.ContainerState*) – a reference to the parent state or None
- **State.state_element_attrs** (*list*) – List of strings/attribute names that point on state element type specific dictionaries which the state hold's as attributes.

property active

Property for the _active field

add_input_data_port(kwargs)**

add_outcome(kwargs)**

add_output_data_port(kwargs)**

add_semantic_data(kwargs)**

change_state_id(state_id=None)

Changes the id of the state to a new id

If no state_id is passed as parameter, a new state id is generated.

Parameters

state_id (*str*) – The new state id of the state

Returns

check_child_validity(child)

Check validity of passed child object

The method is called by state child objects (outcomes, data ports) when these are initialized or changed. The method checks the type of the child and then checks its validity in the context of the state.

Parameters

child (*object*) – The child of the state that is to be tested

Return bool validity, str message

validity is True, when the child is valid, False else. message gives more information especially if the child is not valid

check_input_data_type()

Check the input data types of the state

Checks all input data ports if the handed data is not of the specified type and generate an error logger message with details of the found type conflict.

check_output_data_type()

Check the output data types of the state

Checks all output data ports if the handed data is not of the specified type and generate an error logger message with details of the found type conflict.

property concurrency_queue

Property for the _concurrency_queue field

property core_element_id**static create_output_dictionary_for_state(*state*)**

Return empty output dictionary for a state

Parameters

state – the state of which the output data is determined

Returns

the output data of the target state

property description

Property for the _description field

destroy(*recursive*)

Removes all the state elements.

Parameters

recursive – Flag wether to destroy all state elements which are removed

property file_system_path

Provides the path in the file system where the state is stored

The method returns None if the state was not stored, before,

Return type

str

Returns

the path on the file system where the state is stored

property final_outcome

Property for the _final_outcome field

finalize(*outcome=None*)

Finalize state

This method is called when the run method finishes

Parameters

outcome (*rafcon.core.logical_port.Outcome*) – final outcome of the state

Returns

Nothing for the moment

classmethod from_dict(dictionary)

An abstract method each state has to implement.

Parameters

dictionary – the dictionary of parameters a state can be created from

Raises

exceptions.NotImplementedError – in any case

generate_run_id()**get_data_port_by_id(data_port_id)**

Search for the given data port id in the data ports of the state

The method tries to find a data port in the input and output data ports. :param int data_port_id: the unique id of the data port :return: the data port with the searched id or None if it is not found

get_data_port_ids()**get_default_input_values_for_state(state)**

Computes the default input values for a state

Parameters

state (*State*) – the state to get the default input values for

get_io_data_port_id_from_name_and_type(name, data_port_type, throw_exception=True)

Returns the data_port_id of a data_port with a certain name and data port type

Parameters

- **name** – the name of the target data_port
- **data_port_type** – the data port type of the target data port
- **throw_exception** – throw an exception when the data port does not exist

Returns

the data port specified by the name and the type

Raises

exceptions.AttributeError – if the specified data port does not exist in the input or output data ports

get_next_upper_library_root_state()

Get next upper library root state

The method recursively checks state parent states till finding a StateMachine as parent or a library root state. If self is a LibraryState the next upper library root state is searched and it is not handed self.state_copy.

:return library root state (Execution or ContainerState) or None if self is not a library root state or inside of such :rtype rafcon.core.states.library_state.State:

get_number_of_data_flows()

Generate the number of data flows

Return type

int

Returns

The number of data flows for a state. Per default states do not have data flows.

get_number_of_transitions()

Generate the number of transitions

Return type

int

Returns

The number of transitions for a state. Per default states do not have transitions.

get_path(*appendix=None*, *by_name=False*)

Recursively create the path of the state.

The path is generated in bottom up method i.e. from the nested child states to the root state. The method concatenates either State.state_id (always unique) or State.name (maybe not unique but human readable) as state identifier for the path.

Parameters

- **appendix** (*str*) – the part of the path that was already calculated by previous function calls
- **by_name** (*bool*) – The boolean enables name usage to generate the path

Return type

str

Returns

the full path to the root state

get_previously_executed_state()

Calculates the state that was executed before this state

Returns

The last state in the execution history

get_semantic_data(*path_as_list*)

Retrieves an entry of the semantic data.

Parameters

path_as_list (*list*) – The path in the vividict to retrieve the value from

Returns**get_state_machine()**

Get a reference of the state_machine the state belongs to

:rtype rafcon.core.state_machine.StateMachine :return: respective state machine

get_states_statistics(*hierarchy_level*)

Get states statistic tuple

Return type

tuple

Returns

The number of child states. As per default states do not have child states return 1.

get_storage_path(*appendix=None*)

Recursively create the storage path of the state.

The path is generated in bottom up method i.e. from the nested child states to the root state. The method concatenates the concatenation of (State.name and State.state_id) as state identifier for the path.

Parameters

appendix (*str*) – the part of the path that was already calculated by previous function calls

Return type

str

Returns

the full path to the root state

get_temp_file_system_path()

Provides a temporary path

The path is not fully secure because the all state ids are not globally unique.

get_uppermost_library_root_state()

Find state_copy of uppermost LibraryState

Method checks if there is a parent library root state and assigns it to be the current library root state till there is no further parent library root state.

id()**property income**

Returns the Income of the state

Returns

Income of the state

Return type

Income

property input_data

Property for the _input_data field

property input_data_ports

Property for the _input_data_ports field

See Property.

Parameters

input_data_ports (*dict*) – Dictionary that maps *int* data_port_ids onto values of type
rafccon.core.state_elements.data_port.InputDataPort

Raises

- **exceptions.TypeError** – if the input_data_ports parameter has the wrong type
- **exceptions.AttributeError** – if the key of the input dictionary and the id of the data port do not match

property is_root_state**property is_root_state_of_library**

If self is the attribute LibraryState.state_copy of a LibraryState its the library root state and its parent is a LibraryState :return True or False :rtype bool

join()

Waits until the state finished execution.

property name

Property for the _name field

Return type

str

Returns

Name of state

property outcomes

Property for the _outcomes field

The setter-method substitute State._outcomes with a handed dictionary. The method checks if the elements are of the right type and the keys consistent (Outcome.outcome_id==key). The method does check validity of the elements by calling the parent-setter and in case of failure cancel the operation and recover old outcomes.

Returns

Dictionary outcomes[int, [rafcon.core.state_elements.logical_port.Outcome](#)]
that maps int outcome_ids onto values of type Outcome

Return type

dict

property output_data

Property for the _output_data field

property output_data_ports

Property for the _output_data_ports field

The setter-method substitute State._output_data_ports with a handed dictionary. The method checks if the elements are of the right type and the keys consistent (DataPort.data_port_id==key). The method does check validity of the elements by calling the parent-setter and in case of failure cancel the operation and recover old _output_data_ports.

Returns

Dictionary output_data_ports[int, [rafcon.core.state_elements.data_port.OutputDataPort](#)]
that maps int data_port_ids onto values of type OutputDataPort

Return type

dict

property parent

Property for the _parent field

Return type

[rafcon.core.states.container_state.ContainerState](#)

Returns

A ContainState or value None if there is no parent or the parent is [rafcon.core.state_machine.StateMachine](#)

property paused

Checks, whether the paused event is set

property preempted

Checks, whether the preempted event is set

preemptive_wait(*time=None*)

Waiting method which can be preempted

Use this method if you want a state to pause. In contrast to time.sleep(), the pause can be preempted. This method can also be used if you want to have a daemon thread within a preemptive concurrency state. In this case, time has to be set to None and the method waits indefinitely or until it is preempted from outside. :param time: The time in seconds to wait or None (default) for infinity :return: True, if the wait was preempted, False else

recursively_pause_states()

Pause the state

recursively_preempt_states()

Preempt the state

recursively_resume_states()

Resume the state

remove(*state_element, recursive=True, force=False, destroy=True*)

Remove item from state

Parameters

- **state_element** ([StateElement](#)) – State element to be removed
- **recursive** ([bool](#)) – Only applies to removal of state and decides whether the removal should be called recursively on all child states
- **force** ([bool](#)) – if the removal should be forced without checking constraints
- **destroy** ([bool](#)) – a flag that signals that the state element will be fully removed and dis-assembled

remove_data_flows_with_data_port_id(*data_port_id*)

Remove all data flows whose from_key or to_key equals the passed data_port_id

Parameters

data_port_id – the id of a data_port of which all data_flows should be removed, the id can be a input or output data port id

remove_income(kwargs)****remove_input_data_port(**kwargs)****remove_outcome(**kwargs)****remove_outcome_hook(*outcome_id*)**

Hook for adding more logic when removing an outcome

This hook is intended for the use of inherited classed, which can add more functionality if needed. A container state would remove its transitions going the removed outcome here.

Parameters

outcome_id – The id of the outcome that is removed

remove_output_data_port(kwargs)****remove_semantic_data(**kwargs)**

run(*args, **kwargs)
Implementation of the abstract run() method of the `threading.Thread`

Raises
`exceptions.NotImplementedError` – in any case

property run_id
Property for the `_run_id` field

property semantic_data
Property for the `_semantic_data` field

setup_backward_run()

setup_run()
Executes a generic set of actions that has to be called in the run methods of each derived state class.

Raises
`exceptions.TypeError` – if the input or output data are not of type dict

start(execution_history, backward_execution=False, generate_run_id=True)
Starts the execution of the state in a new thread.

Returns

property started
Checks, whether the started event is set

state_element_attrs = ['income', 'outcomes', 'input_data_ports', 'output_data_ports']

property state_execution_status
Property for the `_state_execution_status` field

property state_id
Property for the `_state_id` field

static state_to_dict(state)

to_dict()
Abstract method

This method must be implemented by the deriving classes. It must return a Python dict with all parameters of self, which are needed to create a copy of self.

Returns
A Python dict with all needed parameters of self

Return type
`dict`

update_hash(obj_hash)
Should be implemented by derived classes to update the hash with their data fields

Parameters
`obj_hash` – The hash object (see Python hashlib)

wait_for_interruption(timeout=None)
Wait for any of the events paused or preempted to be set

Parameters

timeout (*float*) – Maximum time to wait, None if infinitely

Returns

True, is an event was set, False if the timeout was reached

Return type

bool

wait_for_unpause(*timeout=None*)

Wait for any of the events started or preempted to be set

Parameters

timeout (*float*) – Maximum time to wait, None if infinitely

Returns

True, is an event was set, False if the timeout was reached

Return type

bool

rafcon.core.states.state.StateExecutionStatus

alias of STATE_EXECUTION_STATE

rafcon.core.states.state.StateType

alias of STATE_TYPE

Storage: storage**storage (in rafcon.core.storage)**

Helper functions to store a statemachine in the local file system and load it from there

rafcon.core.storage.storage.FILE_NAME_META_DATA = 'meta_data.json'

File names for various purposes

rafcon.core.storage.storage.clean_path(*base_path*)

This function cleans a file system path in terms of removing all not allowed characters of each path element. A path element is an element of a path between the path separator of the operating system.

Parameters

base_path – the path to be cleaned

Returns

the clean path

rafcon.core.storage.storage.clean_path_element(*text, max_length=None, separator='_'*)

Replace characters that conflict with a free OS choice when in a file system path.

Parameters

- **text** – the string to be cleaned
- **max_length** – the maximum length of the output string
- **separator** – the separator used for rafcon.core.storage.storage.limit_text_max_length

Returns

`rafcon.core.storage.storage.clean_path_from_deprecated_naming(base_path)`

Checks if the base path includes deprecated characters/format and returns corrected version

The state machine folder name should be according the universal RAFCON path format. In case the state machine path is inside a mounted library_root_path also the library_path has to have this format. The library path is a partial path of the state machine path. This rules are followed to always provide secure paths for RAFCON and all operating systems.

Parameters

`base_path –`

Returns

cleaned base_path

Return type

`str`

`rafcon.core.storage.storage.find_library_dependencies_via_grep(library_root_path,
library_path=None,
library_name=None)`

Find the dependencies of a library via grep

Parameters

- `library_root_path (str)` – the library root path
- `library_path (str)` – the library path
- `library_name (str)` – the library name

:rtype list(str) :return: library dependency paths

`rafcon.core.storage.storage.get_core_data_path(state_path)`

`rafcon.core.storage.storage.get_meta_data_path(state_path)`

`rafcon.core.storage.storage.get_storage_id_for_state(state)`

Calculates the storage id of a state. This ID can be used for generating the file path for a state.

Parameters

`state (rafcon.core.states.state.State)` – state the storage_id should be composed for

`rafcon.core.storage.storage.limit_text_max_length(text, max_length, separator='_')`

Limits the length of a string. The returned string will be the first `max_length/2` characters of the input string plus a separator plus the last `max_length/2` characters of the input string.

Parameters

- `text` – the text to be limited
- `max_length` – the maximum length of the output string
- `separator` – the separator between the first “`max_length`”/2 characters of the input string and the last “`max_length/2`” characters of the input string

Returns

the shortened input string

`rafcon.core.storage.storage.limit_text_to_be_path_element(text, max_length=None, separator='_')`

Replace characters that are not in the valid character set of RAFCON.

Parameters

- `text` – the string to be cleaned

- **max_length** – the maximum length of the output string
- **separator** – the separator used for rafcon.core.storage.storage.limit_text_max_length

Returns

```
rafcon.core.storage.storage.load_data_file(path_of_file)
```

Loads the content of a file by using json.load.

Parameters

path_of_file – the path of the file to load

Returns

the file content as a string

Raises

exceptions.ValueError – if the file was not found

```
rafcon.core.storage.storage.load_state_recursively(parent, state_path=None, dirty_states=[])
```

Recursively loads the state

It calls this method on each sub-state of a container state.

Parameters

- **parent** – the root state of the last load call to which the loaded state will be added
- **state_path** – the path on the filesystem where to find the meta file for the state
- **dirty_states** – a dict of states which changed during loading

Returns

```
rafcon.core.storage.storage.reconnect_data_flow(state_machine)
```

```
rafcon.core.storage.storage.remove_obsolete_folders(states, path)
```

Removes obsolete state machine folders

This function removes all folders in the file system folder *path* that do not belong to the states given by *states*.

Parameters

- **states** (*list*) – the states that should reside in this very folder
- **path** (*str*) – the file system path to be checked for valid folders

```
rafcon.core.storage.storage.save_script_file_for_state_and_source_path(state, state_path_full,  
as_copy=False)
```

Saves the script file for a state to the directory of the state.

The script name will be set to the SCRIPT_FILE constant.

Parameters

- **state** – The state of which the script file should be saved
- **state_path_full** (*str*) – The path to the file system storage location of the state
- **as_copy** (*bool*) – Temporary storage flag to signal that the given path is not the new file_system_path

```
rafcon.core.storage.storage.save_semantic_data_for_state(state, state_path_full)
```

Saves the semantic data in a separate json file.

Parameters

- **state** – The state of which the script file should be saved
- **state_path_full** (*str*) – The path to the file system storage location of the state

```
rafcon.core.storage.storage.save_state_machine_to_path(state_machine, base_path,  
                                                     delete_old_state_machine=False,  
                                                     as_copy=False)
```

Saves a state machine recursively to the file system

The *as_copy* flag determines whether the state machine is saved as copy. If so (*as_copy=True*), some state machine attributes will be left untouched, such as the *file_system_path* or the *dirty_flag*.

Parameters

- **state_machine** (`rafcon.core.state_machine.StateMachine`) – the state_machine to be saved
- **base_path** (*str*) – base_path to which all further relative paths refers to
- **delete_old_state_machine** (*bool*) – Whether to delete any state machine existing at the given path
- **as_copy** (*bool*) – Whether to use a copy storage for the state machine

```
rafcon.core.storage.storage.save_state_recursively(state, base_path, parent_path, as_copy=False)
```

Recursively saves a state to a json file

It calls this method on all its substates.

Parameters

- **state** – State to be stored
- **base_path** – Path to the state machine
- **parent_path** – Path to the parent state
- **as_copy** (*bool*) – Temporary storage flag to signal that the given path is not the new *file_system_path*

Returns

13.1.2 config

```
class rafcon.core.config.Config(logger_object=None)
```

Bases: *ObservableConfig*

Class to hold and load the global state machine configurations.

```
keys_requiring_restart = ()
```

```
load(config_file=None, path=None)
```

Loads the configuration from a specific file

Parameters

- **config_file** – the name of the config file
- **path** – the path to the config file

```
class rafcon.core.config.ObservableConfig(default_config_string, logger_object=None)
```

Bases: *DefaultConfig*, *Observable*

as_dict()

Returns the configuration as dict

Returns

A copy of the whole configuration as dict

Return type

dict

get_config_value(key, default=None)

Overwrites the default behavior of the get_config_value method of the DefaultConfig base class It supports the

Parameters

- **key** – the key to the configuration value
- **default** – what to return if the key is not found

Returns

The value for the given key, if the key was found. Otherwise the default value

is_config_loaded_from_file()

Returns if the configuration values were loaded from a custom file (and are thus not simply initiated from the default config any more).

Returns

a flag indicating if the config was loaded from a file

property keys

keys_requiring_restart = {}

keys_requiring_state_machine_refresh = {}

set_config_value(kwargs)**

Get a specific configuration value

Parameters

- **key** – the key to the configuration value
- **value** – The new value to be set for the given key

13.1.3 custom_exceptions

exception rafcon.core.custom_exceptions.LibraryNotFoundException(message)

Bases: `Exception`

exception rafcon.core.custom_exceptions.LibraryNotFoundSkipException(message)

Bases: `Exception`

exception rafcon.core.custom_exceptions.RecoveryModeException(message, do_delete_item)

Bases: `ValueError`

property do_delete_item

Property for the `do_delete_item` field

13.1.4 constants (in rafcon.core)

13.1.5 global_variable_manager

13.1.6 id_generator

`rafcon.core.id_generator.generate_data_flow_id()`

`rafcon.core.id_generator.generate_data_port_id(used_data_port_ids)`

Create a new and unique data port id

Parameters

`used_data_port_ids` (`list`) – Handled list of ids already in use

Return type

`int`

Returns

`data_port_id`

`rafcon.core.id_generator.generate_outcome_id(used_outcome_ids)`

`rafcon.core.id_generator.generate_script_id()`

`rafcon.core.id_generator.generate_semantic_data_key(used_semantic_keys)`

Create a new and unique semantic data key

Parameters

`used_semantic_keys` (`list`) – Handled list of keys already in use

Return type

`str`

Returns

`semantic_data_id`

`rafcon.core.id_generator.generate_state_machine_id()`

Generates an id for a state machine. It simply uses a global counter that is increased each time. :return: a new state machine id

`rafcon.core.id_generator.generate_state_name_id()`

Generates an id for a state

It simply uses a global counter that is increased each time. It is intended for the name of a new state.

Returns

a new state machine id

`rafcon.core.id_generator.generate_transition_id()`

`rafcon.core.id_generator.global_variable_id_generator(size=10,
chars='ABCDEFGHIJKLMNPQRSTUVWXYZ')`

Create a new and unique global variable id

Generates an id for a global variable. It randomly samples from random ascii uppercase letters size times and concatenates them. If the id already exists it draws a new one.

Parameters

- `size` – the length of the generated keys

- **chars** – the set of characters a sample draws from

```
rafcon.core.id_generator.history_item_id_generator()
```

```
rafcon.core.id_generator.run_id_generator()
```

```
rafcon.core.id_generator.state_id_generator(size=6, chars='ABCDEFGHIJKLMNPQRSTUVWXYZ',  
                                             used_state_ids=None)
```

Create a new and unique state id

Generates an id for a state. It randomly samples from random ascii uppercase letters size times and concatenates them. If the id already exists it draws a new one.

Parameters

- **size** – the length of the generated keys
- **chars** – the set of characters a sample draws from
- **used_state_ids** (*list*) – Handled list of ids already in use

Return type

str

Returns

new_state_id

13.1.7 interface

```
rafcon.core.interface.create_folder_cmd_line(query, default_name=None, default_path=None)
```

Queries the user for a path to be created

Parameters

- **query** (*str*) – Query that asks the user for a specific folder path to be created
- **default_name** (*str*) – Default name of the folder to be created
- **default_path** (*str*) – Path in which the folder is created if the user doesn't specify a path

Returns

Input path from the user or *default_path* if nothing is specified or None if directory could not be created

Return type

str

```
rafcon.core.interface.create_folder_func(query, default_name=None, default_path=None)
```

Queries the user for a path to be created

Parameters

- **query** (*str*) – Query that asks the user for a specific folder path to be created
- **default_name** (*str*) – Default name of the folder to be created
- **default_path** (*str*) – Path in which the folder is created if the user doesn't specify a path

Returns

Input path from the user or *default_path* if nothing is specified or None if directory could not be created

Return type

str

`rafcon.core.interface.open_folder_cmd_line(query, default_path=None)`

Queries the user for a path to open

Parameters

- **query** (`str`) – Query that asks the user for a specific folder path to be opened
- **default_path** (`str`) – Path to use if the user doesn't specify a path

Returns

Input path from the user or `default_path` if nothing is specified or None if path does not exist

Return type

str

`rafcon.core.interface.open_folder_func(query, default_path=None)`

Queries the user for a path to open

Parameters

- **query** (`str`) – Query that asks the user for a specific folder path to be opened
- **default_path** (`str`) – Path to use if the user doesn't specify a path

Returns

Input path from the user or `default_path` if nothing is specified or None if path does not exist

Return type

str

`rafcon.core.interface.save_folder_cmd_line(query, default_name=None, default_path=None)`

Queries the user for a path or file to be saved into

The folder or file has not to be created already and will not be created by this function. The parent directory of folder and file has to exist otherwise the function will return None.

Parameters

- **query** (`str`) – Query that asks the user for a specific folder/file path to be created
- **default_name** (`str`) – Default name of the folder to be created
- **default_path** (`str`) – Path in which the folder is created if the user doesn't specify a path

Returns

Input path from the user or `default_path` if nothing is specified and None if directory does not exist

Return type

str

`rafcon.core.interface.show_notice(notice, custom_buttons=None)`

Shows a notice on the console that has to be acknowledged

Parameters

notice (`str`) – Notice to show to the user

`rafcon.core.interface.show_notice_func(notice, custom_buttons=None)`

Shows a notice on the console that has to be acknowledged

Parameters

notice (`str`) – Notice to show to the user

13.1.8 library_manager

13.1.9 script

`class rafcon.core.script.Script(path=None, filename=None, parent=None)`

Bases: Observable, YAMLObject

A class for representing the script file for all execution states in a state machine.

It inherits from Observable to make a change of its fields observable.

Variables

- `path` – the path where the script resides
- `filename` – the full name of the script file
- `_compiled_module` – the compiled module
- `_script_id` – the id of the script
- `check_path` – a flag to indicate if the path should be checked for existence

`compile_module()`

Builds a temporary module from the script file

Raises

`exceptions.IOError` – if the compilation of the script module failed

`property compiled_module`

Return the compiled module

`execute(state, inputs=None, outputs=None, backward_execution=False)`

Execute the user ‘execute’ function specified in the script

Parameters

- `state` (`ExecutionState`) – the state belonging to the execute function, refers to ‘self’
- `inputs` (`dict`) – the input data of the script
- `outputs` (`dict`) – the output data of the script
- `backward_execution` (`bool`) – Flag whether to run the script in backwards mode

Returns

Return value of the execute script

Return type

`str | int`

`property filename`

Property for the `_filename` field

`property parent`

Property for the `_parent` field

`property path`

`property script`

`set_script_without_compilation(script_text)`

13.1.10 singleton (in rafcon.core)

13.1.11 state_machine

```
class rafcon.core.state_machine.StateMachine(root_state=None, version=None, creation_time=None,
                                              state_machine_id=None)
```

Bases: Observable, JSONObject, Hashable

A class for organizing all main components of a state machine

It inherits from Observable to make a change of its fields observable.

Variables

- **StateMachine.state_machine_id** (`int`) – the id of the state machine
- **StateMachine.root_state** (`rafcon.core.states.state`) – the root state of the state machine
- **StateMachine.base_path** (`str`) – the path, where to save the state machine

acquire_modification_lock(`blocking=True`)

Acquires the modification lock of the state machine

This must be used for all methods, that perform any modifications on the state machine

Parameters

`blocking` (`bool`) – When True, block until the lock is unlocked, then set it to locked

Returns

True if lock was acquired, False else

Return type

`bool`

change_root_state_type(`**kwargs`)

clear_execution_histories(`**kwargs`)

destroy_execution_histories()

property execution_histories

property file_system_path

Property for the _file_system_path field

classmethod from_dict(`dictionary, state_machine_id=None`)

Abstract method

This method must be implemented by the deriving classes. It must return an object of type cls, created from the parameters defined in dictionary. The type of cls is the type of the class the method is called on.

Parameters

`dictionary` (`dict`) – A Python dict with all parameters needed for creating an object of type cls

Returns

An instance of cls

Return type

`cls`

```
get_last_execution_log_filename()
get_modification_lock()
get_state_by_path(path, as_check=False)
join()
    Wait for root state to finish execution
property marked_dirty
    Property for the _marked_dirty field
modification_lock(blocking=True)
    Get modification lock in with() statement
        Parameters
            blocking (bool) – When True, block until the lock is unlocked, then set it to locked
release_modification_lock()
    Releases the acquired state machine modification lock.
property root_state
    Property for the _root_state field
start()
    Starts the execution of the root state.
state_machine_id = None
static state_machine_to_dict(state_machine)
property supports_saving_state_names
to_dict()
    Abstract method
    This method must be implemented by the deriving classes. It must return a Python dict with all parameters
    of self, which are needed to create a copy of self.
        Returns
            A Python dict with all needed parameters of self
        Return type
            dict
update_hash(obj_hash)
    Should be implemented by derived classes to update the hash with their data fields
        Parameters
            obj_hash – The hash object (see Python hashlib)
version = None
```

13.1.12 state_machine_manager

13.2 The GUI: gui

Contents

- *The GUI: gui*
 - *Subpackages of rafcon.gui*
 - *action*
 - *clipboard*
 - *shortcut_manager*
 - *singleton (in rafcon.gui)*

13.2.1 Subpackages of rafcon.gui

MVC Controllers: rafcon.gui.controllers

All controllers of the MVC architecture.

The controllers are the interface between the models (holding the data) and the view (showing the data). They are responsible for the logic.

All controllers of RAFCON inherit from `rafcon.gui.controllers.extended_controller.ExtendedController`, which inherits from the controller class of GTKMVC.

Contents

- *MVC Controllers: rafcon.gui.controllers*
 - *Subpackages of rafcon.gui.controllers*
 - *MainWindowController (in main_window)*
 - *GraphicalEditorController (in graphical_editor_gaphas)*
 - *StateMachinesEditorController (in state_machines_editor)*
 - *ExecutionHistoryTreeController (in execution_history)*
 - *GlobalVariableManagerController (in global_variable_manager)*
 - *LoggingConsoleController (in logging_console)*
 - *LibraryTreeController (in library_tree)*
 - *MenuBarController (in menu_bar)*
 - *ModificationHistoryTreeController (in modification_history)*
 - *StateMachineTreeController (in state_machine_tree)*
 - *StatesEditorController (in states_editor)*
 - *ToolBarController (in tool_bar)*

- *TopToolBarController* (in *top_tool_bar*)
- *StateIconController* (in *state_icons*)
- *UndockedWindowController* (in *undocked_window*)

Subpackages of rafcon.gui.controllers

MVC State Editor Controllers (rafcon.gui.controllers.state_editor)

Contents

- *MVC State Editor Controllers* (*rafcon.gui.controllers.state_editor*)
 - *LinkageOverviewController* (in *linkage_overview*)
 - *ScopedVariableListController* (in *scoped_variable_list*)
 - *SourceEditorController* (in *source_editor*)
 - *StateDataFlowsListController* (in *data_flows*)
 - *DescriptionEditorController* (in *description_editor*)
 - *StateEditorController* (in *state_editor*)
 - *StateOutcomesListController* (in *outcomes*)
 - *StateOverviewController* (in *overview*)
 - *StateTransitionsListController* (in *transitions*)

LinkageOverviewController (in linkage_overview)

```
class rafcon.gui.controllers.state_editor.linkage_overview.LinkageOverviewController(model,
view)
```

Bases: *ExtendedController*, *ModelMT*

ScopedVariableListController (in scoped_variable_list)

```
class rafcon.gui.controllers.state_editor.scoped_variable_list.ScopedVariableListController(model,
view)
```

Bases: *ListViewController*

Controller handling the scoped variable list

Parameters

- **model** (*rafcon.gui.models.state.StateModel*) – The state model, holding state data.
- **view** (*rafcon.gui.views.scoped_variables_list.ScopedVariablesListView*) – The GTK view showing the list of scoped variables.

CORE_ELEMENT_CLASS

alias of *ScopedVariable*

DATA_TYPE_NAME_STORAGE_ID = 1**DEFAULT_VALUE_STORAGE_ID = 2****ID_STORAGE_ID = 3****MODEL_STORAGE_ID = 4****NAME_STORAGE_ID = 0****apply_new_scoped_variable_default_value(path, new_default_value_str)**

Applies the new default value of the scoped variable defined by path

Parameters

- **path (str)** – The path identifying the edited variable
- **new_default_value_str (str)** – New default value as string

apply_new_scoped_variable_name(path, new_name)

Applies the new name of the scoped variable defined by path

Parameters

- **path (str)** – The path identifying the edited variable
- **new_name (str)** – New name

apply_new_scoped_variable_type(path, new_variable_type_str)

Applies the new data type of the scoped variable defined by path

Parameters

- **path (str)** – The path identifying the edited variable
- **new_variable_type_str (str)** – New data type as str

static get_new_list_store()**on_add(widget, data=None)**

Create a new scoped variable with default values

on_right_click_menu()

An abstract method called after right click events

paste_action_callback(*event, **kwargs)

Callback method for paste action

The method trigger the clipboard paste of the list of scoped variables in the clipboard or in case this list is empty and there are other port types selected in the clipboard it will trigger the paste with convert flag. The convert flag will cause the insertion of scoped variables with the same names, data types and default values the objects of differing port type (in the clipboard) have.

register_actions(shortcut_manager)

Register callback methods for triggered actions

Parameters

- shortcut_manager (rafcon.gui.shortcut_manager.ShortcutManager)** – Shortcut Manager Object holding mappings between shortcuts and actions.

register_view(*view*)

Called when the View was registered

reload_scoped_variables_list_store()

Reloads the scoped variable list store from the data port models

remove_core_element(*model*)

Remove respective core element of handed scoped variable model

Parameters

model (`ScopedVariableModel`) – Scoped variable model which core element should be removed

Returns**scoped_variables_changed(*model, prop_name, info*)****SourceEditorController (in `source_editor`)****class rafcon.gui.controllers.state_editor.source_editor.SourceEditorController(*model, view*)**

Bases: `EditorController, AbstractExternalEditor`

Controller handling the source editor in Execution States.

:param :param rafcon.gui.views.source_editor.SourceEditorView view: The GTK view showing the source editor.

apply_clicked(*button*)

Triggered when the Apply button in the source editor is clicked.

static format_error_string(*message*)**get_file_name()**

Implements the abstract method of the ExternalEditor class.

load_and_set_file_content(*file_system_path*)

Implements the abstract method of the ExternalEditor class.

register_view(*view*)

Called when the View was registered

Can be used e.g. to connect signals. Here, this implements a convenient feature that observes if thread problems are possible by destroying a controller before being fully initialized.

save_file_data(*path*)

Implements the abstract method of the ExternalEditor class.

set_editor_lock(*lock*)

Implements the abstract method of the ExternalEditor class.

property source_text

```
tmp_file = '/tmp/rafcon-docs/2407/tmphwky1fyg/file_to_get_pylinted.py'
```

StateDataFlowsListController (in `data_flows`)

```
class rafcon.gui.controllers.state_editor.data_flows.StateDataFlowsEditorController(model,  
view)
```

Bases: `ExtendedController`

register_actions(`shortcut_manager`)

Register callback methods for triggered actions

Parameters

`shortcut_manager` (`rafcon.gui.shortcut_manager.ShortcutManager`) –

register_view(`view`)

Called when the View was registered

Can be used e.g. to connect signals. Here, the destroy signal is connected to close the application

toggled_button(`button, name=None`)

```
class rafcon.gui.controllers.state_editor.data_flows.StateDataFlowsListController(model,  
view)
```

Bases: `LinkageListController`

Controller handling the view of transitions of the ContainerStateModel

This `gtkmvc3.Controller` class is the interface between the GTK widget view `gui.views.data_flow.DataFlowListView` and the transitions of the `gui.models.state.ContainerStateModel`. Changes made in the GUI are written back to the model and vice versa.

Parameters

- `model` (`rafcon.gui.models.ContainerStateModel`) – The container state model containing the data
- `view` (`rafcon.gui.views.DataFlowListView`) – The GTK view showing the data flows as a table

CORE_ELEMENT_CLASS

alias of `DataFlow`

`FROM_STATE_STORAGE_ID = 1`

`ID_STORAGE_ID = 0`

`IS_EXTERNAL_STORAGE_ID = 5`

`MODEL_STORAGE_ID = 11`

`TO_KEY_STORAGE_ID = 4`

`TO_STATE_STORAGE_ID = 3`

`after_notification_of_parent_or_state`(`model, prop_name, info`)

`after_notification_of_parent_or_state_from_lists`(`model, prop_name, info`)

`before_notification_of_parent_or_state`(`model, prop_name, info`)

Set the no update flag to avoid updates in between of a state removal.

destroy()

Recursively destroy all Controllers

The method remove all controllers, which calls the destroy method of the child controllers. Then, all registered models are relieved and and the widget hand by the initial view argument is destroyed.

find_free_and_valid_data_flows(depend_to_state_id=None)

free_to_port_external = None

free_to_port_internal = None

from_port_external = None

from_port_internal = None

on_add(button, info=None)

An abstract add method for a respective new core element and final selection of those

on_combo_changed_from_key(widget, path, text)**on_combo_changed_from_state(widget, path, text)****on_combo_changed_to_key(widget, path, text)****on_combo_changed_to_state(widget, path, text)****on_focus(widget, data=None)****on_right_click_menu()**

An abstract method called after right click events

register_view(view)

Called when the View was registered

remove_core_element(model)

Remove respective core element of handed data flow model

Parameters

- **model** ([DataFlowModel](#)) – Data Flow model which core element should be removed

Returns**update(initiator='Unknown')**

rafcn.gui.controllers.state_editor.data_flows.**find_free_keys**(model)

rafcn.gui.controllers.state_editor.data_flows.**get_key_combos**(ports, keys_store, not_key=None)

rafcn.gui.controllers.state_editor.data_flows.**get_state_model**(state_m, state_id)

rafcn.gui.controllers.state_editor.data_flows.**update_data_flows**(model, data_flow_dict, tree_dict_combos)

Updates data flow dictionary and combo dictionary of the widget according handed model.

Parameters

- **model** – model for which the data_flow_dict and tree_dict_combos should be updated
- **data_flow_dict** – dictionary that holds all internal and external data-flows and those respective row labels

- **tree_dict_combos** – dictionary that holds all internal and external data-flow-adaptation-combos

Returns**DescriptionEditorController (in description_editor)**

```
class rafcon.gui.controllers.state_editor.description_editor.DescriptionEditorController(model,  
view)
```

Bases: *EditorController*

Controller handling the description editor of States.

:param :param rafcon.gui.views.source_editor.SourceEditorView view: The GTK view showing the source editor.

on_focus_out(*args, **kwargs)

register_actions(shortcut_manager)

Register callback methods for triggered actions

Parameters

shortcut_manager ([rafcon.gui.shortcut_manager.ShortcutManager](#)) – Shortcut Manager Object holding mappings between shortcuts and actions.

register_view(view)

Called when the View was registered

Can be used e.g. to connect signals. Here, this implements a convenient feature that observes if thread problems are possible by destroying a controller before being fully initialized.

scroll_to_bottom(widget, data=None)

property source_text

StateEditorController (in state_editor)

```
class rafcon.gui.controllers.state_editor.state_editor.StateEditorController(model, view)
```

Bases: *ExtendedController*

Controller handles the organization of the Logic-Data oriented State-Editor. Widgets concerning logic flow (outcomes and transitions) are grouped in the Logic Linkage expander. Widgets concerning data flow (data-ports and data-flows) are grouped in the data linkage expander.

Parameters

model ([rafcon.gui.models.state.StateModel](#)) – The state model

register_view(view)

Called when the View was registered

Can be used e.g. to connect signals. Here, the destroy signal is connected to close the application

rename()

show_content_changed(model, prop_name, info)

state_destruction(model, prop_name, info)

Close state editor when state is being destructed

StateOutcomesListController (in outcomes)

```
class rafcon.gui.controllers.state_editor.outcomes.StateOutcomesEditorController(model,  
view)
```

Bases: *ExtendedController*

paste_action_callback(*event, **kwargs)

Callback method for paste action

register_actions(shortcut_manager)

Register callback methods for triggered actions

Parameters

shortcut_manager (*rafcon.gui.shortcut_manager.ShortcutManager*) –

register_view(view)

Called when the View was registered

Can be used e.g. to connect signals. Here, the destroy signal is connected to close the application

```
class rafcon.gui.controllers.state_editor.outcomes.StateOutcomesListController(model, view)
```

Bases: *ListViewController*

The controller handles the outcomes of one respective state

The controller allows to add and remove outcomes as well as to add, remove and to modify the related transition.

The related transition can be set to a sibling-state, to the state it self or to a outcome of the parent. Hereby the transition also can switch from pointing to an outcome or to a state. It react to changes in the state's respective outcomes-list, transitions-list or change of parent-state and use additionally the focus change to update after a modification (e.g. the focus change updates if not observed state-names change).

CORE_ELEMENT_CLASS

alias of *Outcome*

CORE_PARENT_STORAGE_ID = 5

CORE_STORAGE_ID = 4

ID_STORAGE_ID = 0

MODEL_STORAGE_ID = 6

NAME_STORAGE_ID = 1

apply_new_outcome_name(path, new_name)

Apply the newly entered outcome name it is was changed

Parameters

- **path** (*str*) – The path string of the renderer

- **new_name** (*str*) – Newly entered outcome name

destroy()

Recursively destroy all Controllers

The method remove all controllers, which calls the destroy method of the child controllers. Then, all registered models are relieved and and the widget hand by the initial view argument is destroyed.

on_add(button, info=None)

An abstract add method for a respective new core element and final selection of those

on_right_click_menu()

An abstract method called after right click events

on_to_outcome_edited(renderer, path, new_outcome_identifier)

Connects the outcome with a transition to the newly set outcome

Parameters

- **renderer** (*Gtk.CellRendererText*) – The cell renderer that was edited
- **path** (*str*) – The path string of the renderer
- **new_outcome_identifier** (*str*) – An identifier for the new outcome that was selected

on_to_state_edited(renderer, path, new_state_identifier)

Connects the outcome with a transition to the newly set state

Parameters

- **renderer** (*Gtk.CellRendererText*) – The cell renderer that was edited
- **path** (*str*) – The path string of the renderer
- **new_state_identifier** (*str*) – An identifier for the new state that was selected

outcomes_changed(model, prop_name, info)**register_view(view)**

Called when the View was registered

Can be used e.g. to connect signals. Here, the destroy signal is connected to close the application

remove_core_element(model)

Remove respective core element of handed outcome model

Parameters

model (*OutcomeModel*) – Outcome model which core element should be removed

Returns**update(initiator='Unknown')****update_internal_data_base()****update_list_store()**

StateOverviewController (in overview)

```
class rafcon.gui.controllers.state_editor.overview.StateOverviewController(model, view,
with_is_start_state_check_box=False)
```

Bases: *ExtendedController*

Controller handling the view of properties/attributes of the ContainerStateModel

This *design_patterns.mvc.controller.Controller* class is the interface between the GTK widget *view.gui.views.source_editor.SourceEditorView* and the properties of the *gui.models.state.StateModel*. Changes made in the GUI are written back to the model and vice versa.

Parameters

- **model** (`rafcon.gui.models.StateModel`) – The state model containing the data
- **view** (`rafcon.gui.views.SourceEditorView`) – The GTK view showing the data as a table

```
change_name(new_name)
change_type(widget, model=None, info=None)
check_for_enter(entry, event)
static get_allowed_state_classes(state)
notify_is_start(model, prop_name, info)
notify_name_change(model, prop_name, info)
on_focus_out(entry, event)
on_toggle_is_start_state(button)
on_toggle_show_content(checkbox)
register_view(view)
    Called when the View was registered
    Can be used e.g. to connect signals. Here, the destroy signal is connected to close the application
    Parameters
        view (rafcon.gui.views.state_editor.overview.StateOverviewView) – A state overview view instance
rename()
show_content_changed(model, prop_name, info)
```

StateTransitionsListController (in transitions)

```
class rafcon.gui.controllers.state_editor.transitions.StateTransitionsEditorController(model,
    view)
Bases: ExtendedController
register_actions(shortcut_manager)
    Register callback methods for triggered actions
    Parameters
        shortcut_manager (rafcon.gui.shortcut_manager.ShortcutManager) –
register_view(view)
    Called when the View was registered
    Can be used e.g. to connect signals. Here, the destroy signal is connected to close the application
toggled_button(button, name=None)
```

```
class rafcon.gui.controllers.state_editor.transitions.StateTransitionsListController(model,
view)
```

Bases: LinkageListController

Controller handling the view of transitions of the ContainerStateModel

This gtkmvc3.Controller class is the interface between the GTK widget view gui.views.transitions.TransitionListView and the transitions of the gui.models.state.ContainerStateModel. Changes made in the GUI are written back to the model and vice versa.

Parameters

- **model** (*rafcon.gui.models.ContainerStateModel*) – The container state model containing the data
- **view** (*rafcon.gui.views.TransitionListView*) – The GTK view showing the transitions as a table

CORE_ELEMENT_CLASS

alias of *Transition*

FROM_OUTCOME_STORAGE_ID = 2

FROM_STATE_STORAGE_ID = 1

ID_STORAGE_ID = 0

IS_EXTERNAL_STORAGE_ID = 5

MODEL_STORAGE_ID = 9

TO_OUTCOME_STORAGE_ID = 4

TO_STATE_STORAGE_ID = 3

after_notification_of_parent_or_state_from_lists(model, prop_name, info)

Activates the update after update if outcomes, transitions or states list has been changed.

after_notification_state(model, prop_name, info)

before_notification_of_parent_or_state(model, prop_name, info)

Set the no update flag to avoid updates in between of a state removal.

destroy()

Recursively destroy all Controllers

The method remove all controllers, which calls the destroy method of the child controllers. Then, all registered models are relieved and the widget hand by the initial view argument is destroyed.

static get_possible_combos_for_transition(trans, model, self_model, is_external=False)

The function provides combos for a transition and its respective

Parameters

- **trans** –
- **model** –
- **self_model** –
- **is_external** –

Returns

on_add(button, info=None)

An abstract add method for a respective new core element and final selection of those

on_combo_changed_from_outcome(widget, path, text)

on_combo_changed_from_state(widget, path, text)

on_combo_changed_to_outcome(widget, path, text)

on_combo_changed_to_state(widget, path, text)

on_focus(widget, data=None)

on_right_click_menu()

An abstract method called after right click events

register_view(view)

Called when the View was registered

remove_core_element(model)

Remove respective core element of handed transition model

Parameters

model ([TransitionModel](#)) – Transition model which core element should be removed

Returns

update(initiator='Unknown')

MVC Controller Utils ([rafcon.gui.controllers.utils](#))

Contents

- *MVC Controller Utils (rafcon.gui.controllers.utils)*
 - *EditorController (in editor)*
 - *ExtendedController (in extended_controller)*
 - *SingleWidgetWindowController (in single_widget_window)*

[EditorController \(in editor\)](#)

```
class rafcon.gui.controllers.utils.editor.EditorController(model, view,
                                                               observed_method='script_text')
```

Bases: *ExtendedController*

Controller handling the text editor for States.

:param :param rafcon.gui.views.source_editor.EditorView view: The GTK view showing the editor.

after_notification_of_script_text_was_changed(model, prop_name, info)

apply_clicked(button)

Triggered when the Apply-Shortcut in the editor is triggered.

cancel_clicked(button)

Triggered when the Cancel-Shortcut in the editor is triggered

Resets the code in the editor to the last-saved code.

code_changed(source)

Apply checks and adjustments of the TextBuffer and TextView after every change in buffer.

The method re-apply the tag (style) for the buffer. It avoids changes while editable-property set to False which are caused by a bug in the GtkSourceView2. GtkSourceView2 is the default used TextView widget here. The text buffer is reset after every change to last stored source-text by a respective work around which suspends any generation of undo items and avoids a recursive call of the method set_enabled by observing its while_in_set_enabled flag.

Parameters

source (*TextBuffer*) –

Returns**register_actions(shortcut_manager)**

Register callback methods for triggered actions

Parameters

shortcut_manager ([rafcon.gui.shortcut_manager.ShortcutManager](#)) – Shortcut Manager Object holding mappings between shortcuts and actions.

register_view(view)

Called when the View was registered

Can be used e.g. to connect signals. Here, this implements a convenient feature that observes if thread problems are possible by destroying a controller before being fully initialized.

set_script_text(text)**property source_text****ExtendedController (in `extended_controller`)****class rafcon.gui.controllers.utils.extended_controller.`ExtendedController`(model, view)**

Bases: Controller

add_controller(key, controller)

Add child controller

The passed controller is registered as child of self. The register_actions method of the child controller is called, allowing the child controller to register shortcut callbacks.

Parameters

- **key** – Name of the controller (unique within self), to later access it again
- **controller** ([ExtendedController](#)) – Controller to be added as child

connect_signal(widget, signal, callback)

destroy()

Recursively destroy all Controllers

The method remove all controllers, which calls the destroy method of the child controllers. Then, all registered models are relieved and and the widget hand by the initial view argument is destroyed.

disconnect_all_signals()**get_child_controllers()**

Returns a list with all registered child controllers

Returns

List of child controllers

Return type

list

get_controller(key)

Return the child controller registered with the name key

Parameters

key – The name of the controller

Returns

The controller

Return type

ExtendedController

get_controller_by_path(ctrl_path, with_print=False)**get_root_window()****observe_model(model)**

Make this model observable within the controller

The method also keeps track of all observed models, in order to be able to relieve them later on.

Parameters

model (*gtkmvc3.Model*) – The model to be observed

property parent

Return the parent controller for which this controller is registered as a child.

Returns

The parent controller

Return type

ExtendedController

register_actions(shortcut_manager)

Register callback methods for triggered actions in all child controllers.

Parameters

shortcut_manager (*rafcon.gui.shortcut_manager.ShortcutManager*) – Shortcut Manager Object holding mappings between shortcuts and actions.

register_view(view)

Called when the View was registered

Can be used e.g. to connect signals. Here, this implements a convenient feature that observes if thread problems are possible by destroying a controller before being fully initialized.

relieve_all_models()

Relieve all registered models

The method uses the set of registered models to relieve them.

relieve_model(*model*)

Do no longer observe the model

The model is also removed from the internal set of tracked models.

Parameters

model (*gtkmvc3.Model*) – The model to be relieved

remove_controller(*controller*)

Remove child controller and destroy it

Removes all references to the child controller and calls `destroy()` on the controller.

Parameters

controller (*str* / *ExtendedController*) – Either the child controller object itself or its registered name

Returns

Whether the controller was existing

Return type

bool

unregister_actions(*shortcut_manager*)**SingleWidgetWindowController (in single_widget_window)**

```
class rafcon.gui.controllers.utils.single_widget_window.SingleWidgetWindowController(model,
                                                                 view,
                                                                 ctrl_class,
                                                                 *args,
                                                                 **kwargs)
```

Bases: *ExtendedController*

Controller handling the view of properties/attributes of ...

register_view(*view*)

Called when the View was registered

Can be used e.g. to connect signals. Here, the `destroy` signal is connected to close the application

MainWindowController (in main_window)**GraphicalEditorController (in graphical_editor_gaphas)****StateMachinesEditorController (in state_machines_editor)****ExecutionHistoryTreeController (in execution_history)**

```
class rafcon.gui.controllers.execution_history.ExecutionHistoryTreeController(model=None,
view=None)
```

Bases: *ExtendedController*

Controller handling the execution history.

Parameters

- **model** (*rafcon.gui.models.state_machine_manager.StateMachineManagerModel*) – The state machine manager model, holding data regarding state machines.
- **view** (*rafcon.gui.views.execution_history.ExecutionHistoryTreeView*) – The GTK View showing the execution history tree.
- **state_machine_manager** (*rafcon.core.state_machine_manager.StateMachineManager*) –

HISTORY_ITEM_STORAGE_ID = 1

TOOL_TIP_STORAGE_ID = 2

TOOL_TIP_TEXT = 'Right click for more details\nMiddle click for external more detailed viewer\nDouble click to select corresponding state'

append_string_to_menu(*popup_menu*, *menu_item_string*)

clean_history(*widget*, *event=None*)

Triggered when the ‘Clean History’ button is clicked.

Empties the execution history tree by adjusting the start index and updates tree store and view.

destroy()

Recursively destroy all Controllers

The method remove all controllers, which calls the destroy method of the child controllers. Then, all registered models are relieved and the widget hand by the initial view argument is destroyed.

execution_history_focus(*model*, *prop_name*, *info*)

Arranges to put execution-history widget page to become top page in notebook when execution starts and stops and resets the boolean of modification_history_was_focused to False each time this notification are observed.

get_history_item_for_tree_iter(*child_tree_iter*)

Hands history item for tree iter and compensate if tree item is a dummy item

Parameters

- **child_tree_iter** (*Gtk.TreeIter*) – Tree iter of row

Rtype *rafcon.core.execution.execution_history.HistoryItem*

Return history tree item

insert_concurrent_execution_histories(*parent*, *concurrent_execution_histories*)

Adds the child execution histories of a concurrency state.

Parameters

- **parent** (*Gtk.TreeItem*) – the parent to add the next history item to
- **concurrent_execution_histories** (*list[ExecutionHistory]*) – a list of all child execution histories

Returns**insert_execution_history**(*parent*, *execution_history*, *is_root=False*)

Insert a list of history items into a the tree store

If there are concurrency history items, the method is called recursively.

Parameters

- **parent** (*Gtk.TreeItem*) – the parent to add the next history item to
- **execution_history** (*ExecutionHistory*) – all history items of a certain state machine execution
- **is_root** (*bool*) – Whether this is the root execution history

insert_history_item(*parent*, *history_item*, *description*, *dummy=False*)

Enters a single history item into the tree store

Parameters

- **parent** (*Gtk.TreeItem*) – Parent tree item
- **history_item** (*HistoryItem*) – History item to be inserted
- **description** (*str*) – A description to be added to the entry
- **dummy** (*None*) – Whether this is just a dummy entry (wrapper for concurrency items)

Returns

Inserted tree item

Return type*Gtk.TreeItem***mouse_click**(*widget*, *event=None*)

Triggered when mouse click is pressed in the history tree. The method shows all scoped data for an execution step as tooltip or fold and unfold the tree by double-click and select respective state for double clicked element.

notification_selected_sm_changed(*model*, *prop_name*, *info*)

If a new state machine is selected, make sure expansion state is stored and tree updated

notification_sm_changed(*model*, *prop_name*, *info*)

Remove references to non-existing state machines

open_selected_history_separately(*widget*, *event=None*)**register_view**(*view*)

Called when the View was registered

Can be used e.g. to connect signals. Here, this implements a convenient feature that observes if thread problems are possible by destroying a controller before being fully initialized.

reload_history(*widget*, *event=None*)

Triggered when the ‘Reload History’ button is clicked.

update()

rebuild the tree view of the history item tree store :return:

GlobalVariableManagerController (in `global_variable_manager`)

```
class rafcon.gui.controllers.global_variable_manager.GlobalVariableManagerController(model,
view)
```

Bases: `ListViewController`

Controller handling the Global Variable Manager

The controller enables to edit, add and remove global variable to the global variable manager by a tree view. Every global variable is accessible by its key which is in the tree view equivalent with its name and in the methods it is `gv_name`. This Controller inherit and use rudimentary methods of the `ListViewController` (therefore it introduce the `ID_STORAGE_ID` class attribute) and avoids to use the selection methods of those which need a `MODEL_STORAGE_ID` (there is no global variable model) and a state machine selection (is model based). Therefore the register view is only called for the extended controller. Because of this and the fact that `name = key` for a global variable `ID_STORAGE_ID`, `NAME_STORAGE_ID` and `MODEL_STORAGE_ID` are all equal.

Parameters

- `model` (`rafcon.gui.models.global_variable_manager.GlobalVariableManagerModel`) – The Global Variable Manager Model
- `view` (`rafcon.gui.views.global_variable_editor.GlobalVariableEditorView`) – The GTK view showing the list of global variables.

Variables

- `global_variable_counter` (`int`) – Counter for global variables to ensure unique names for new global variables.
- `list_store` (`Gtk.ListStore`) – A gtk list store storing the rows of data of respective global variables in.

`DATA_TYPE_AS_STRING_STORAGE_ID = 1`

`ID_STORAGE_ID = 0`

`IS_LOCKED_AS_STRING_STORAGE_ID = 3`

`MODEL_STORAGE_ID = 0`

`NAME_STORAGE_ID = 0`

`VALUE_AS_STRING_STORAGE_ID = 2`

`apply_new_global_variable_name(path, new_gv_name)`

Change global variable name/key according handed string

Updates the global variable name only if different and already in list store.

Parameters

- `path` – The path identifying the edited global variable tree view row, can be str, int or tuple.
- `new_gv_name` (`str`) – New global variable name

`apply_new_global_variable_type(path, new_data_type_as_string)`

Change global variable value according handed string

Updates the global variable data type only if different.

Parameters

- **path** – The path identifying the edited global variable tree view row, can be str, int or tuple.
- **new_data_type_as_string** (*str*) – New global variable data type as string

apply_new_global_variable_value(*path, new_value_as_string*)

Change global variable value according handed string

Updates the global variable value only if new value string is different to old representation.

Parameters

- **path** – The path identifying the edited global variable tree view row, can be str, int or tuple.
- **new_value_as_string** (*str*) – New global variable value as string

assign_notification_from_gvm(*model, prop_name, info*)

Handles gtkmvc3 notification from global variable manager

Calls update of whole list store in case new variable was added. Avoids to run updates without reasonable change. Holds tree store and updates row elements if is-locked or global variable value changes.

global_variable_is_editable(*gv_name, intro_message='edit'*)

Check whether global variable is locked

Parameters

- **gv_name** (*str*) – Name of global variable to be checked
- **intro_message** (*str*) – Message which is used form a useful logger error message if needed

Returns**on_add**(*widget, data=None*)

Create a global variable with default value and select its row

Triggered when the add button in the global variables tab is clicked.

on_lock(*widget, data=None*)

Locks respective selected core element

on_unlock(*widget, data=None*)

Locks respective selected core element

register_actions(*shortcut_manager*)

Register callback methods for triggered actions

Parameters

- **shortcut_manager** ([rafcon.gui.shortcut_manager.ShortcutManager](#)) – Shortcut Manager Object holding mappings between shortcuts and actions.

register_view(*view*)

Called when the View was registered

remove_core_element(*model*)

Remove respective core element of handed global variable name

Parameters

- **model** (*str*) – String that is the key/gv_name of core element which should be removed

Returns

update_global_variables_list_store()

Updates the global variable list store

Triggered after creation or deletion of a variable has taken place.

LoggingConsoleController (in logging_console)

```
class rafcon.gui.controllers.logging_console.LoggingConsoleController(model, view)
```

Bases: *ExtendedController*

Controller handling the updates and modifications of the logging console.

Parameters

- **rafcon.gui.models.config_model.ConfigModel** – Gui config model holding and observing the global gui config.
- **view(rafcon.gui.views.logging_console.LoggingConsoleView)** – The GTK view showing the logging messages.

add_clear_menu_item(widget, menu)

destroy()

Recursively destroy all Controllers

The method remove all controllers, which calls the destroy method of the child controllers. Then, all registered models are relieved and the widget hand by the initial view argument is destroyed.

model_changed(model, prop_name, info)

React to configuration changes

Update internal hold enable state, propagates it to view and refresh the text buffer.

print_filtered_buffer()

print_message(message, log_level, new=True)

register_view(view)

Called when the View was registered

Can be used e.g. to connect signals. Here, this implements a convenient feature that observes if thread problems are possible by destroying a controller before being fully initialized.

update_filtered_buffer()

LibraryTreeController (in library_tree)

```
class rafcon.gui.controllers.library_tree.LibraryTreeController(model, view, find_usages=False)
```

Bases: *ExtendedController*

ID_STORAGE_ID = 0

ITEM_STORAGE_ID = 1

LIB_KEY_STORAGE_ID = 4

LIB_PATH_STORAGE_ID = 2

```
OS_PATH_STORAGE_ID = 3
TOOL_TIP_STORAGE_ID = 3
static convert_if_human_readable(s)
    Converts a string to format which is more human readable
extract_library_properties_from_selected_row()
    Extracts properties library_os_path, library_path, library_name and tree_item_key from tree store row
generate_right_click_menu(kind='library')
get_menu_item_text(menu_item)
insert_button_clicked(widget, as_template=False)
insert_rec(parent, library_key, library_item, library_path, library_root_path=None)
```

Parameters

- **parent** –
- **library_key** (*str*) –
- **library_item** –
- **library_path** (*str*) –
- **library_root_path** (*str*) –

Returns

```
menu_item_add_library_root_clicked(widget)
menu_item_find_usages_clicked(widget)
menu_item_relocate_libraries_or_root_clicked(menu_item)
    Relocate library after request second confirmation
menu_item_remove_libraries_or_root_clicked(menu_item)
    Removes library from hard drive after request second confirmation
menu_item_rename_libraries_or_root_clicked(menu_item)
    Rename library after request second confirmation
model_changed(model, prop_name, info)
mouse_click(widget, event=None)
on_drag_begin(widget, context)
    replace drag icon
```

Parameters

- **widget** –
- **context** –

```
on_drag_data_get(widget, context, data, info, time)
    dragged state is inserted and its state_id sent to the receiver
```

Parameters

- **widget** –

- **context** –
- **data** – SelectionData: contains state_id
- **info** –
- **time** –

open_button_clicked(widget)

open_library_as_state_machine()

open_run_button_clicked(widget)

redo_expansion_state()

register_view(view)
 Called when the View was registered
 Can be used e.g. to connect signals. Here, this implements a convenient feature that observes if thread problems are possible by destroying a controller before being fully initialized.

select_library_tree_element_of_lib_tree_path(lib_tree_path)

select_library_tree_element_of_library_state_model(state_m)

store_expansion_state()

substitute_as_library_clicked(widget, keep_name=True)

substitute_as_template_clicked(widget, keep_name=True)

update()

MenuBarController (in menu_bar)

```
class rafcon.gui.controllers.menu_bar.MenuBarController(state_machine_manager_model, view,
                                                       shortcut_manager, sm_execution_engine)
```

Bases: *ExtendedController*

Controller handling the Menu Bar

The class to trigger all the actions, available in the menu bar.

Parameters

- **state_machine_manager_model** (*rafcon.gui.models.state_machine_manager.StateMachineManagerModel*) – The state machine manager model, holding data regarding state machines. Should be exchangeable.
- **view** (*rafcon.gui.views.main_window.MainWindowView*) – The GTK View showing the Menu Bar and Menu Items.

add_callback_to_shortcut_manager(action, callback)

Helper function to add an callback for an action to the shortcut manager. :param action: the action to add a shortcut for :param callback: the callback if the action is executed :return:

call_action_callback(*callback_name*, **args*, ***kwargs*)

Wrapper for action callbacks

Returns True after executing the callback. This is needed in order to prevent the shortcut from being passed on to the system. The callback methods itself cannot return True, as they are also used with idle_add, which would call the method over and over again. :param str *callback_name*: The name of the method to call :param *args*: Any remaining parameters, which are passed on to the callback method :return: True

check_edit_menu_items_status(*widget*)

connect_button_to_function(*view_index*, *button_state*, *function*)

Connect callback to a button :param *view_index*: the index of the button in the view :param *button_state*: the state of the button the function should be connected to :param *function*: the function to be connected :return:

static create_logger_warning_if_shortcuts_are_overwritten_by_menu_bar()

data_flow_mode_toggled_shortcut(**args*, ***kwargs*)

destroy()

Recursively destroy all Controllers

The method remove all controllers, which calls the destroy method of the child controllers. Then, all registered models are relieved and and the widget hand by the initial view argument is destroyed.

static on_about_activate(*widget*, *data=None*)

on_add_state_activate(*widget*, *method=None*, **arg*)

static on_add_transitions_from_closest_sibling_state_active(*widget*, *data=None*)

static on_add_transitions_to_closest_sibling_state_active(*widget*, *data=None*)

static on_add_transitions_to_parent_state_active(*widget*, *data=None*)

on_backward_step_activate(*widget*, *data=None*)

on_bake_state_machine_activate(*widget*, *data=None*, *force=False*)

on_config_value_changed(*config_m*, *prop_name*, *info*)

Callback when a config value has been changed

Parameters

- **config_m** (*ConfigModel*) – The config model that has been changed
- **prop_name** (*str*) – Should always be ‘config’
- **info** (*dict*) – Information e.g. about the changed config key

on_copy_selection_activate(*widget*, *data=None*)

on_cut_selection_activate(*widget*, *data=None*)

static on_data_flow_mode_toggled(*widget*, *data=None*)

on_delete_activate(*widget*, *data=None*)

on_delete_check_sm_modified()

on_delete_check_sm_running()

```

on_delete_event(widget, event, data=None, force=False)

on_destroy(widget, data=None)

on_escape_key_press_event_leave_full_screen(widget, event)

on_expert_view_activate(widget, data=None)

on_full_screen_activate(*args)
    function to display the currently selected state machine in full screen mode :param args: :return:

on_full_screen_deactivate()

on_full_screen_mode_toggled(*args)

on_grid_toggled(widget, data=None)

on_group_states_activate(widget, data=None)

static on_layout_state_machine(*args, **kwargs)

static on_menu_preferences_activate(widget, data=None)

on_new_activate(widget=None, data=None)

static on_open_activate(widget=None, data=None, path=None)

static on_open_library_state_separately_activate(widget, data=None, cursor_position=None)

on_paste_clipboard_activate(widget, data=None)

on_pause_activate(widget, data=None)

on_quit_activate(widget, data=None, force=False)

on_redo_activate(widget, data=None)

on_refresh_all_activate(widget, data=None, force=False)

static on_refresh_libraries_activate()

on_refresh_selected_activate(widget, data=None, force=False)

on_run_only_selected_state_activate(widget, data=None)

on_run_selected_state_activate(widget, data=None)

on_run_to_selected_state_activate(widget, data=None)

on_save_activate(widget, data=None, delete_old_state_machine=False)

on_save_as_activate(widget=None, data=None, path=None)

on_save_as_copy_activate(widget=None, data=None, path=None)

static on_save_selected_state_as_activate(widget=None, data=None, path=None)

on_search_activate(widget, data=None, cursor_position=None)

static on_show_aborted_preempted_toggled(widget, data=None)

```

```
static on_show_data_flows_toggled(widget, data=None)
static on_show_data_values_toggled(widget, data=None)
static on_show_transitions_toggled(widget, data=None)
on_start_activate(widget, data=None)
on_start_from_selected_state_activate(widget, data=None)
on_step_into_activate(widget, data=None)
on_step_mode_activate(widget, data=None)
on_step_out_activate(widget, data=None)
on_step_over_activate(widget, data=None)
on_stop_activate(widget, data=None)
static on_substitute_library_with_template_activate(widget=None, data=None)
static on_substitute_selected_state_activate(widget=None, data=None, path=None)
on_toggle_full_screen_mode(*args, **kwargs)
static on_toggle_is_start_state_active(widget, data=None)
on_undo_activate(widget, data=None)
on_ungroup_state_activate(widget, data=None)
refresh_shortcuts()
register_actions(shortcut_manager)
    Register callback methods for triggered actions
    Parameters
        shortcut_manager (rafcon.gui.shortcut\_manager.ShortcutManager) – Shortcut
        Manager Object holding mappings between shortcuts and actions.
register_view(view)
    Called when the View was registered
show_aborted_preempted(*args, **kwargs)
show_data_flows_toggled_shortcut(*args, **kwargs)
show_data_values_toggled_shortcut(*args, **kwargs)
show_transitions_toggled_shortcut(*args, **kwargs)
unregister_view()
    import log Unregister all registered functions to a view element :return:
```

ModificationHistoryTreeController (in modification_history)

```
class rafcon.gui.controllers.modification_history.ModificationHistoryTreeController(model,
view)
```

Bases: *ExtendedController*

static get_color_active(active)

new_change(model, method_name, instance, info, history_id, active, parent_tree_item, parameters,
tool_tip=None)

on_cursor_changed(widget)

on_redo_button_clicked(widget, event=None)

on_reset_button_clicked(widget, event=None)

on_toggle_mode(widget, event=None)

on_toggle_tree_folded(widget, event=None)

on_undo_button_clicked(widget, event=None)

redo(key_value, modifier_mask, **kwargs)

Redo for selected state-machine if no state-source-editor is open and focused in states-editor-controller.

Returns

True if a redo was performed, False if focus on source-editor.

Return type

`bool`

register()

Change the state machine that is observed for new selected states to the selected state machine. :return:

register_actions(shortcut_manager)

Register callback methods for triggered actions

Parameters

`shortcut_manager (rafcon.gui.shortcut_manager.ShortcutManager) –`

register_view(view)

Called when the View was registered

Can be used e.g. to connect signals. Here, this implements a convenient feature that observes if thread problems are possible by destroying a controller before being fully initialized.

state_machine_manager_notification(model, property, info)

undo(key_value, modifier_mask, **kwargs)

Undo for selected state-machine if no state-source-editor is open and focused in states-editor-controller.

Returns

True if a undo was performed, False if focus on source-editor.

Return type

`bool`

update(model, prop_name, info)

The method updates the history (a Gtk.TreeStore) which is the model of respective TreeView. Its functionality is strongly depends on a consistent history-tree hold by a ChangeHistory-Class.

StateMachineTreeController (in state_machine_tree)**class** rafcon.gui.controllers.state_machine_tree.StateMachineTreeController(*model, view*)

Bases: TreeViewController

Controller handling the state machine tree.

Parameters

- **model** (*rafcon.gui.models.state_machine_manager.StateMachineManagerModel*) – The state machine manager model, holding data regarding state machines. Should be exchangeable.
- **view** (*rafcon.gui.views.state_machine_tree.StateMachineTreeView*) – The GTK view showing the state machine tree.

CORE_ELEMENT_CLASSalias of [State](#)**ID_STORAGE_ID** = 1**MODEL_STORAGE_ID** = 3**NAME_STORAGE_ID** = 0**STATE_PATH_STORAGE_ID** = 4**TYPE_NAME_STORAGE_ID** = 2**action_signal**(*model, prop_name, info*)**assign_notification_selection**(*state_machine_m, signal_name, signal_msg*)**get_row_iter_for_state_model**(*state_model*)**get_state_machine_selection()**

Getter state machine selection

Returns

selection object, filtered set of selected states

Return typerafcon.gui.selection.Selection, [set](#)**insert_and_update_recursively**(*parent_iter, state_model, with_expand=False*)

Insert and/or update the handed state model in parent tree store element iterator

Parameters

- **parent_iter** – Parent tree store iterator the insert should be performed in
- **state_model** ([StateModel](#)) – Model of state that has to be insert and/or updated
- **with_expand** ([bool](#)) – Trigger to expand tree

Returns**mouse_click**(*widget, event=None*)**paste_action_callback**(**event, **kwargs*)

Callback method for paste action

redo_expansion_state(*ignore_not_existing_rows=False*)

 Considers the tree to be collapsed and expand into all tree item with the flag set True

register()

 Change the state machine that is observed for new selected states to the selected state machine.

register_actions(*shortcut_manager*)

 Register callback methods for triggered actions

Parameters

shortcut_manager ([rafcon.gui.shortcut_manager.ShortcutManager](#)) – Shortcut Manager Object holding mappings between shortcuts and actions.

register_view(*view*)

 Called when the view was registered

remove_tree_children(*child_tree_iter*)

selection_changed(*widget, event=None*)

 Notify state machine about tree view selection

show_content(*state_model*)

 Check state machine tree specific show content flag.

Is returning true if the upper most library state of a state model has a enabled show content flag or if there is no library root state above this state.

Parameters

state_model ([rafcon.gui.models.abstract_state.AbstractStateModel](#)) – The state model to check

state_action_signal(*model, prop_name, info*)

state_machine_manager_notification(*model, property, info*)

state_machine_notification(*model, property, info*)

state_meta_update(*model, prop_name, info*)

states_update(*model, prop_name, info*)

states_update_before(*model, prop_name, info*)

store_expansion_state()

update(*changed_state_model=None, with_expand=False*)

 Checks if all states are in tree and if tree has states which were deleted

Parameters

- **changed_state_model** – Model that row has to be updated
- **with_expand** – The expand flag for the tree

update_tree_store_row(*state_model*)

StatesEditorController (in states_editor)**class rafcon.gui.controllers.states_editor.StatesEditorController(model, view)**Bases: *ExtendedController*

Controller handling the states editor

Parameters

- **model** (*rafcon.gui.models.state_machine_manager.StateMachineManagerModel*) – The state machine manager model, holding data regarding state machines.
- **view** (*rafcon.gui.views.states_editor.StatesEditorView*) – The GTK view showing state editor tabs.

Variables

- **tabs** – Currently open State Editor tabs.
- **closed_tabs** – Previously opened, non-deleted State Editor tabs.

activate_state_tab(state_m)

Opens the tab for the specified state model

The tab with the given state model is opened or set to foreground.

Parameters**state_m** – The desired state model (the selected state)**add_state_editor(state_m)**

Triggered whenever a state is selected.

Parameters**state_m** – The selected state model.**close_all_pages()**

Closes all tabs of the states editor

close_page(state_identifier, delete=True)

Closes the desired page

The page belonging to the state with the specified state_identifier is closed. If the deletion flag is set to False, the controller of the page is stored for later usage.

Parameters

- **state_identifier** – Identifier of the page's state
- **delete** – Whether to delete the controller (deletion is necessary if the state is deleted)

close_pages_for_specific_sm_id(sm_id)

Closes all tabs of the states editor for a specific sm_id

property current_state_machine_m**get_current_state_m()**

Returns the state model of the currently open tab

get_page_of_state_m(state_m)

Return the identifier and page of a given state model

Parameters

state_m – The state model to be searched

Returns

page containing the state and the state_identifier

get_state_identifier(state_m)

get_state_identifier_for_page(page)

Returns the state identifier for a given page

notify_state_name_change(model, prop_name, info)

Checks whether the name of a state was changed and change the tab label accordingly

on_close_clicked(widget, page_num)

on_config_value_changed(config_m, prop_name, info)

Callback when a config value has been changed

Parameters

- **config_m (ConfigModel)** – The config model that has been changed
- **prop_name (str)** – Should always be ‘config’
- **info (dict)** – Information e.g. about the changed config key

on_switch_page(notebook, page_pointer, page_num, user_param1=None)

Update state machine and state selection when the active tab was changed

When the state editor tab switches (e.g. when a tab is closed), this callback causes the state machine tab corresponding to the activated state editor tab to be opened and the corresponding state to be selected.

This is disabled for now, as the might be irritating for the user

on_tab_close_clicked(event, state_m)

Triggered when the states-editor close button is clicked

Closes the tab.

Parameters

state_m – The desired state model (the selected state)

on_toggle_sticky_clicked(event, state_m)

Callback for the “toggle-sticky-check-button” emitted by custom TabLabel widget.

prepare_destruction()

register_actions(shortcut_manager)

Register callback methods for triggered actions

Parameters

shortcut_manager (rafcon.gui.shortcut_manager.ShortcutManager) – Shortcut Manager Object holding mappings between shortcuts and actions.

register_view(view)

Called when the View was registered

Can be used e.g. to connect signals. Here, this implements a convenient feature that observes if thread problems are possible by destroying a controller before being fully initialized.

reload_style()

Closes all tabs and reopens only the current tab in the new style.

Returns**rename_selected_state(key_value, modifier_mask, **kwargs)**

Callback method for shortcut action rename

Searches for a single selected state model and open the according page. Page is created if it is not existing. Then the rename method of the state controller is called.

Parameters

- **key_value** –

- **modifier_mask** –

root_state_changed(model, property, info)**script_text_changed(text_buffer, state_m)**

Update gui elements according text buffer changes

Checks if the dirty flag needs to be set and the tab label to be updated.

Parameters

- **text_buffer** (*TextBuffer*) – Text buffer of the edited script

- **state_m** ([rafcon.gui.models.state.StateModel](#)) – The state model related to the text buffer

Returns**selection_notification(state_machine_m, property, info)**

If a single state is selected, open the corresponding tab

state_action_signal(model, prop_name, info)**state_machine_manager_notification(model, property, info)**

Triggered whenever a new state machine is created, or an existing state machine is selected.

state_machines_del_notification(model, prop_name, info)

Relive models of closed state machine

state_machines_set_notification(model, prop_name, info)

Observe all open state machines and their root states

update_tab_label(state_m)

Update all tab labels

Parameters

- **state_m** ([rafcon.state_machine.states.state.State](#)) – State model who's tab label is to be updated

`rafcon.gui.controllers.states_editor.create_button(toggle, font_size, icon_code,
release_callback=None, *additional_parameters)`

`rafcon.gui.controllers.states_editor.create_sticky_button(callback, *additional_parameters)`

`rafcon.gui.controllers.states_editor.create_tab_close_button(callback, *additional_parameters)`

```
rafcon.gui.controllers.states_editor.create_tab_header(title, close_callback, sticky_callback,  
*additional_parameters)  
  
rafcon.gui.controllers.states_editor.set_tab_label_texts(label, state_m, unsaved_changes=False)
```

ToolBarController (in tool_bar)

```
class rafcon.gui.controllers.tool_barToolBarController(state_machine_manager_model, view)
```

Bases: *ExtendedController*

The class to trigger all the action, available in the tool bar.

Parameters

- **state_machine_manager_model** (*rafcon.gui.models.state_machine_manager.StateMachineManagerModel*) – The state machine manager model, holding data regarding state machines. Should be exchangeable.

- **view** –

```
on_button_bake_state_machine_clicked(widget, data=None)
```

```
on_button_layout_state_machine(widget, data=None)
```

```
on_button_new_clicked(widget, data=None)
```

```
on_button_open_clicked(widget, data=None)
```

```
on_button_refresh_clicked(widget, data=None)
```

```
on_button_refresh_libs_clicked(widget, data=None)
```

```
on_button_refresh_selected_clicked(widget, data=None)
```

```
on_button_save_clicked(widget, data=None)
```

```
register_actions(shortcut_manager)
```

Register callback methods for triggered actions

Parameters

```
shortcut_manager (rafcon.gui.shortcut_manager.ShortcutManager) –
```

```
register_view(view)
```

Called when the View was registered

TopToolBarController (in top_tool_bar)

```
class rafcon.gui.controllers.top_tool_barTopToolBarController(state_machine_manager_model,  
view, top_level_window)
```

Bases: *ExtendedController*

The class to trigger all the actions available in the top tool bar.

Parameters

- **state_machine_manager_model** (*rafcon.gui.models.state_machine_manager.StateMachineManagerModel*) – The state machine manager model, holding data regarding state machines. Should be exchangeable.

- **view**(*rafcon.gui.views.top_tool_bar.TopToolBarView*) – The GTK View showing the top tool bar buttons.
- **top_level_window** – The top level window containing the top tool bar.

on_maximize_button_clicked(*widget, data=None*)

on_minimize_button_clicked(*widget, data=None*)

register_view(*view*)

Called when the View was registered

update_maximize_button()

```
class rafcon.gui.controllers.top_tool_bar.TopToolBarUndockedWindowController(state_machine_manager_model,
                                                                           view, re-
                                                                           dock_method)
```

Bases: *TopToolBarController*

Controller handling the top tool bar in the un-docked windows.

In this controller, the close button in the top tool bar is hidden.

on_redock_button_clicked(*widget, event=None*)

Triggered when the redock button in any window is clicked.

Calls the corresponding redocking function of the open window.

register_view(*view*)

Called when the View was registered

StateIconController (in state_icons)

```
class rafcon.gui.controllers.state_icons.StateIconController(model=None, view=None,
                                                               shortcut_manager=None)
```

Bases: *ExtendedController*

on_drag_begin(*widget, context*)

replace drag icon

Parameters

- **widget** –
- **context** –

on_drag_data_get(*widget, context, data, info, time*)

dragged state is inserted and its state_id sent to the receiver

Parameters

- **widget** –
- **context** –
- **data** – SelectionData: contains state_id
- **info** –
- **time** –

on_drag_end(widget, context)

if the drag is finished, all icons are unselected

Parameters

- **widget** –
- **context** –

on_mouse_click(widget, event)

state insertion on mouse click

Parameters

- **widget** –
- **event (Gdk.Event)** – mouse click event

on_mouse_motion(widget, event)

selection on mouse over

Parameters

- **widget** –
- **event (Gdk.Event)** – mouse motion event

register_view(view)

Called when the View was registered

Can be used e.g. to connect signals. Here, this implements a convenient feature that observes if thread problems are possible by destroying a controller before being fully initialized.

UndockedWindowController (in `undocked_window`)

class rafcon.gui.controllers.undocked_window.UndockedWindowController(state_machine_manager_model, view, redock_method)

Bases: *ExtendedController*

Controller handling the un-docked windows

Parameters

- **state_machine_manager_model (rafcon.gui.models.state_machine_manager.StateMachineManagerModel)** – The state machine manager model, holding data regarding state machines. Should be exchangeable.
- **view (rafcon.gui.views.undocked_window.UndockedWindowView)** – The GTK View showing the separate window

hide_window()

MVC Models: rafcon.gui.models

This package contains all models of the MVC architecture.

The models hold the data for, which are shown in the views. These models typically hold an element of the core, which is observable. For example, the StateModel holds a reference to a State class object from the core. If the core element changes, the model recognizes these changes and forwards the notification to the controllers, if they observe the changed model property.

Contents

- *MVC Models: rafcon.gui.models*
 - *AbstractStateModel (in abstract_state)*
 - *StateModel (in state)*
 - *ContainerStateModel (in container_state)*
 - *LibraryStateModel (in library_state)*
 - *StateElementModel (in state_element)*
 - *TransitionModel (in transition)*
 - *DataFlowModel (in data_flow)*
 - *DataPortModel (in data_port)*
 - *ScopedVariableModel (in scoped_variable)*
 - *Selection (in selection)*
 - *LogicalPortModel (in logical_port)*
 - *StateMachineModel (in state_machine)*
 - *ModificationsHistoryModel (in modification_history)*
 - *AutoBackupModel (in auto_backup)*
 - *StateMachineManagerModel (in state_machine_manager)*
 - *GlobalVariableManagerModel (in global_variable_manager)*
 - *LibraryManagerModel (in library_manager)*
 - *StateMachineExecutionEngineModel (in state_machine_execution_engine)*

AbstractStateModel (in abstract_state)

```
class rafcon.gui.models.abstract_state.AbstractStateModel(state, parent=None, meta=None)
```

Bases: *MetaModel*, *Hashable*

This is an abstract class serving as base class for state models

The model class is part of the MVC architecture. It holds the data to be shown (in this case a state).

Parameters

- **state** – The state to be managed which can be any derivative of `rafcon.core.states.state.State`.
- **parent (AbstractStateModel)** – The state to be managed

- **meta** (`rafcon.utils.vividict.Vividict`) – The meta data of the state

property action_signal

action_signal_triggered(model, prop_name, info)

This method notifies the parent state and child state models about complex actions

child_model_changed(notification_overview)

copy_meta_data_from_state_m(source_state_m)

Dismiss current meta data and copy meta data from given state model

The meta data of the given state model is used as meta data for this state. Also the meta data of all state elements (data ports, outcomes, etc.) is overwritten with the meta data of the elements of the given state.

Parameters

source_state_m – State model to load the meta data from

property core_element

property destruction_signal

get_data_port_m(data_port_id)

Searches and returns the model of a data port of a given state

The method searches a port with the given id in the data ports of the given state model. If the state model is a container state, not only the input and output data ports are looked at, but also the scoped variables.

Parameters

data_port_id – The data port id to be searched

Returns

The model of the data port or None if it is not found

get_input_data_port_m(data_port_id)

Returns the input data port model for the given data port id

Parameters

data_port_id – The data port id to search for

Returns

The model of the data port with the given id

get_observable_action_signal()

get_observable_destruction_signal()

get_observable_income()

get_observable_input_data_ports()

get_observable_is_start()

get_observable_meta_signal()

get_observable_outcomes()

get_observable_output_data_ports()

get_observable_state()

get_outcome_m(*outcome_id*)

Returns the outcome model for the given outcome id

Parameters

outcome_id – The outcome id to search for

Returns

The model of the outcome with the given id

get_output_data_port_m(*data_port_id*)

Returns the output data port model for the given data port id

Parameters

data_port_id – The data port id to search for

Returns

The model of the data port with the given id

get_state_machine_m(*two_factor_check=True*)

Get respective state machine model

Get a reference of the state machine model the state model belongs to. As long as the root state model has no direct reference to its state machine model the state machine manager model is checked respective model.

Return type

rafccon.gui.models.state_machine.StateMachineModel

Returns

respective state machine model

property hierarchy_level**property income****property input_data_ports****property is_about_to_be_destroyed_recursively****property is_start****load_meta_data(*path=None*)**

Load meta data of state model from the file system

The meta data of the state model is loaded from the file system and stored in the meta property of the model. Existing meta data is removed. Also the meta data of all state elements (data ports, outcomes, etc) are loaded, as those stored in the same file as the meta data of the state.

This is either called on the `__init__` of a new state model or if a state model for a container state is created, which then calls `load_meta_data` for all its children.

Parameters

path (*str*) – Optional file system path to the meta data file. If not given, the path will be derived from the state's path on the filesystem

Returns

if meta data file was loaded True otherwise False

Return type

bool

meta_changed(model, prop_name, info)

This method notifies the parent state about changes made to the meta data

property meta_signal

model_changed(model, prop_name, info)

This method notifies parent state about changes made to the state

property outcomes

property output_data_ports

property parent

prepare_destruction(recursive=True)

Prepares the model for destruction

Recursively un-registers all observers and removes references to child models

set_observable_action_signal(value)

set_observable_destruction_signal(value)

set_observable_income(value)

set_observable_input_data_ports(value)

set_observable_is_start(value)

set_observable_meta_signal(value)

set_observable_outcomes(value)

set_observable_output_data_ports(value)

set_observable_state(value)

property state

state_counter = 0

store_meta_data(copy_path=None)

Save meta data of state model to the file system

This method generates a dictionary of the meta data of the state together with the meta data of all state elements (data ports, outcomes, etc.) and stores it on the filesystem. Secure that the store meta data method is called after storing the core data otherwise the last_stored_path is maybe wrong or None. The copy path is considered to be a state machine file system path but not the current one but e.g. of a as copy saved state machine. The meta data will be stored in respective relative state folder in the state machine hierarchy. This folder has to exist. Dues the core elements of the state machine has to be stored first.

Parameters

copy_path (*str*) – Optional copy path if meta data is not stored to the file system path of state machine

update_hash(obj_hash)

Should be implemented by derived classes to update the hash with their data fields

Parameters

obj_hash – The hash object (see Python hashlib)

update_is_start()

Updates the `is_start` property of the state

A state is a start state, if it is the root state, it has no parent, the parent is a LibraryState or the state's `state_id` is identical with the `ContainerState.start_state_id` of the ContainerState it is within.

update_meta_data_hash(`obj_hash`)

Should be implemented by derived classes to update the hash with their meta data fields

Parameters

obj_hash – The hash object (see Python `hashlib`)

rafcon.gui.models.abstract_state.get_state_model_class_for_state(`state`)

Determines the model required for the given state class

Parameters

state – Instance of a state (`ExecutionState`, `BarrierConcurrencyState`, ...)

Returns

The model class required for holding such a state instance

rafcon.gui.models.abstract_state.mirror_y_axis_in_vividict_element(`vividict`, `key`)**StateModel (in state)**

```
class rafcon.gui.models.state.StateModel(state, parent=None, meta=None, load_meta_data=True,  
expected_future_models=None)
```

Bases: `AbstractStateModel`

This model class manages a State, for the moment only `ExecutionStates`

The model class is part of the MVC architecture. It holds the data to be shown (in this case a state).

Parameters

- **state** (`rafcon.core.states.state.State`) – The state to be managed
- **parent** (`AbstractStateModel`) – The state to be managed
- **meta** (`rafcon.utils.vividict.Vividict`) – The meta data of the state
- **expected_future_models** (`__buildIn__.set`) – Existing models for new core elements

add_missing_model(`model_list_or_dict`, `core_elements_dict`, `model_name`, `model_class`, `model_key`)

Adds one missing model

The method will search for the first core-object out of `core_object_dict` not represented in the list or dict of models handed by `model_list_or_dict`, adds it and returns without continue to search for more objects which maybe are missing in `model_list_or_dict` with respect to the `core_object_dict`.

Parameters

- **model_list_or_dict** – could be a list or dictionary of one model type
- **core_elements_dict** – dictionary of one type of core-elements (`rafcon.core`)
- **model_name** – prop_name for the core-element hold by the model, this core-element is covered by the model
- **model_class** – model-class of the elements that should be insert
- **model_key** – if `model_list_or_dict` is a dictionary the key is the id of the respective element (e.g. ‘`state_id`’)

Returns

True, is a new model was added, False else

Return type

`bool`

```
expected_future_models = None
get_cause_and_affected_model_list(model)
get_model_info(model)
model_changed(model, prop_name, info)
```

This method notifies the model lists and the parent state about changes

The method is called each time, the model is changed. This happens, when the state itself changes or one of its children (outcomes, ports) changes. Changes of the children cannot be observed directly, therefore children notify their parent about their changes by calling this method. This method then checks, what has been changed by looking at the method that caused the change. In the following, it notifies the list in which the change happened about the change. E.g. one input data port changes its name. The model of the port observes itself and notifies the parent (i.e. the state model) about the change by calling this method with the information about the change. This method recognizes that the method “`modify_input_data_port`” caused the change and therefore triggers a notify on the list if input data port models. “`notify_before`” is used as trigger method when the changing function is entered and “`notify_after`” is used when the changing function returns. This changing function in the example would be “`modify_input_data_port`”.

Parameters

- **model** – The model that was changed
- **prop_name** – The property that was changed
- **info** – Information about the change (e.g. the name of the changing function)

`re_initiate_model_list(model_list_or_dict, core_objects_dict, model_name, model_class, model_key)`

Recreate model list

The method re-initiate a handed list or dictionary of models with the new dictionary of core-objects.

Parameters

- **model_list_or_dict** – could be a list or dictionary of one model type
- **core_objects_dict** – new dictionary of one type of core-elements (rafcon.core)
- **model_name** – prop_name for the core-element hold by the model, this core-element is covered by the model
- **model_class** – model-class of the elements that should be insert
- **model_key** – if model_list_or_dict is a dictionary the key is the id of the respective element (e.g. ‘state_id’)

Returns

`remove_additional_model(model_list_or_dict, core_objects_dict, model_name, model_key, destroy=True)`

Remove one unnecessary model

The method will search for the first model-object out of model_list_or_dict that represents no core-object in the dictionary of core-objects handed by core_objects_dict, remove it and return without continue to search for more model-objects which maybe are unnecessary, too.

Parameters

- **model_list_or_dict** – could be a list or dictionary of one model type
- **core_objects_dict** – dictionary of one type of core-elements (rafcon.core)
- **model_name** – prop_name for the core-element hold by the model, this core-element is covered by the model
- **model_key** – if model_list_or_dict is a dictionary the key is the id of the respective element (e.g. ‘state_id’)

Returns

```
remove_specific_model(model_list_or_dict, core_element, model_key=None, recursive=True,  
destroy=True)
```

```
update_models(model, name, info)
```

This method is always triggered when the core state changes

It keeps the following models/model-lists consistent: input-data-port models output-data-port models outcome models

ContainerStateModel (in container_state)

```
class rafcon.gui.models.container_state.ContainerStateModel(container_state, parent=None,  
meta=None, load_meta_data=True,  
expected_future_models=None)
```

Bases: *StateModel*

This model class manages a ContainerState

The model class is part of the MVC architecture. It holds the data to be shown (in this case a container state).

Parameters

container_state (*ContainerState*) – The container state to be managed

```
copy_meta_data_from_state_m(source_state_m)
```

Dismiss current meta data and copy meta data from given state model

In addition to the state model method, also the meta data of container states is copied. Then, the meta data of child states are recursively copied.

Parameters

source_state_m – State model to load the meta data from

```
property data_flows
```

```
get_cause_and_affected_model_list(model)
```

```
get_data_flow_m(data_flow_id)
```

Searches and return the data flow model with the given in the given container state model

Parameters

data_flow_id – The data flow id to be searched

Returns

The model of the data flow or None if it is not found

get_data_port_m(*data_port_id*)

Searches and returns the model of a data port of a given state

The method searches a port with the given id in the data ports of the given state model. If the state model is a container state, not only the input and output data ports are looked at, but also the scoped variables.

Parameters

data_port_id – The data port id to be searched

Returns

The model of the data port or None if it is not found

get_observable_data_flows()**get_observable_scoped_variables()****get_observable_states()****get_observable_transitions()****get_scoped_variable_m(*data_port_id*)**

Returns the scoped variable model for the given data port id

Parameters

data_port_id – The data port id to search for

Returns

The model of the scoped variable with the given id

get_transition_m(*transition_id*)

Searches and return the transition model with the given in the given container state model

Parameters

transition_id – The transition id to be searched

Returns

The model of the transition or None if it is not found

group_states(*model, prop_name, info*)**insert_meta_data_from_models_dict(*source_models_dict, notify_logger_method*)****model_changed(*model, prop_name, info*)**

This method notifies the model lists and the parent state about changes

The method is called each time, the model is changed. This happens, when the state itself changes or one of its children (states, transitions, data flows) changes. Changes of the children cannot be observed directly, therefore children notify their parent about their changes by calling this method. This method then checks, what has been changed by looking at the model that is passed to it. In the following it notifies the list in which the change happened about the change. E.g. one child state changes its name. The model of that state observes itself and notifies the parent (i.e. this state model) about the change by calling this method with the information about the change. This method recognizes that the model is of type StateModel and therefore triggers a notify on the list of state models. “notify_before” is used as trigger method when the changing function is entered and “notify_after” is used when the changing function returns. This changing function in the example would be the setter of the property name. :param model: The model that was changed :param prop_name: The property that was changed :param info: Information about the change (e.g. the name of the changing function)

prepare_destruction(*recursive=True*)

Prepares the model for destruction

Recursively un-registers all observers and removes references to child models. Extends the destroy method of the base class by child elements of a container state.

property scoped_variables**set_observable_data_flows(*value*)****set_observable_scoped_variables(*value*)****set_observable_states(*value*)****set_observable_transitions(*value*)****property states****store_meta_data(*copy_path=None*)**

Store meta data of container states to the filesystem

Recursively stores meta data of child states. For further insides read the description of also called respective super class method.

Parameters

copy_path (*str*) – Optional copy path if meta data is not stored to the file system path of state machine

substitute_state(*model, prop_name, info*)**property transitions****ungroup_state(*model, prop_name, info*)****update_child_is_start()**

Updates the *is_child* property of its child states

update_child_models(*_, name, info*)

This method is always triggered when the state model changes

It keeps the following models/model-lists consistent: transition models data-flow models state models scoped variable models

update_hash(*obj_hash*)

Should be implemented by derived classes to update the hash with their data fields

Parameters

obj_hash – The hash object (see Python hashlib)

update_meta_data_hash(*obj_hash*)

Should be implemented by derived classes to update the hash with their meta data fields

Parameters

obj_hash – The hash object (see Python hashlib)

LibraryStateModel (in library_state)

```
class rafcon.gui.models.library_state.LibraryStateModel(state, parent=None, meta=None,
load_meta_data=True)
```

Bases: *AbstractStateModel*

This model class manages a LibraryState

The model class is part of the MVC architecture. It holds the data to be shown (in this case a state).

Parameters

state (*rafcon.core.states.library_state.LibraryState*) – The state to be managed

copy_meta_data_from_state_m(source_state_m)

Dismiss current meta data and copy meta data from given state model

The meta data of the given state model is used as meta data for this state. Also the meta data of all state elements (data ports, outcomes, etc.) is overwritten with the meta data of the elements of the given state.

Parameters

source_state_m – State model to load the meta data from

enforce_generation_of_state_copy_model()

This enforce a load of state copy model without considering meta data

initiate_library_root_state_model()

property is_about_to_be_destroyed_recursively

model_changed(model, prop_name, info)

This method notifies parent state about changes made to the state

prepare_destruction(recursive=True)

Prepares the model for destruction

Recursively un-registers all observers and removes references to child models

recursive_generate_models(load_meta_data)

show_content()

Check if content of library is to be shown

Content is shown, if the uppermost state's meta flag “show_content” is True and the library hierarchy depth (up to MAX_VISIBLE_LIBRARY_HIERARCHY level) is not to high.

Returns

Whether the content is to be shown

Return type

bool

state_copy = None

update_hash(obj_hash)

Should be implemented by derived classes to update the hash with their data fields

Parameters

obj_hash – The hash object (see Python hashlib)

StateElementModel (in state_element)**class** rafcon.gui.models.state_element.StateElementModel(*parent, meta=None*)

Bases: MetaModel, Hashable

This model class serves as base class for all models within a state model (ports, connections)

Each state element model has a parent, meta and temp data. If observes itself and informs the parent about changes.

Parameters

- **parent** (rafcon.gui.models.abstract_state.AbstractStateModel) – The state model of the state element
- **meta** (rafcon.utils.vividict.Vividict) – The meta data of the state element model

property core_element

Return the core element represented by this model

Returns

core element of the model

Return type

rafcon.core.state_elements.state_element.StateElement

property destruction_signal**get_observable_destruction_signal()****get_observable_meta_signal()****get_state_machine_m()****meta_changed(model, prop_name, info)**

This method notifies the parent state about changes made to the meta data

property meta_signal**model_changed(model, prop_name, info)**

This method notifies the parent state about changes made to the state element

property parent

Getter for the parent state model of the state element

Returns

None if parent is not defined, else the model of the parent state

Return type

rafcon.gui.models.abstract_state.AbstractState

prepare_destruction()

Prepares the model for destruction

Unregisters the model from observing itself.

set_observable_destruction_signal(value)**set_observable_meta_signal(value)**

update_hash(*obj_hash*)

Should be implemented by derived classes to update the hash with their data fields

Parameters

obj_hash – The hash object (see Python hashlib)

TransitionModel (in transition)**class rafcon.gui.models.transition.TransitionModel(*transition, parent, meta=None*)**

Bases: *StateElementModel*

This model class manages a Transition

Parameters

- **transition** (*rafcon.core.transition.Transition*) – The transition to be wrapped
- **parent** (*rafcon.gui.models.abstract_state.AbstractStateModel*) – The state model of the state element
- **meta** (*rafcon.utils.vividict.Vividict*) – The meta data of the state element model

property core_element

Return the core element represented by this model

Returns

core element of the model

Return type

rafcon.core.state_elements.state_element.StateElement

get_observable_transition()**model_changed(*model, prop_name, info*)**

This method notifies the parent state about changes made to the state element

prepare_destruction()

Prepares the model for destruction

Unregisters the model from observing itself.

set_observable_transition(*value*)**property transition****rafcn.gui.models.transition.mirror_waypoints(*vividict*)****DataFlowModel (in data_flow)****class rafcon.gui.models.data_flow.DataFlowModel(*data_flow, parent, meta=None*)**

Bases: *StateElementModel*

This model class manages a DataFlow

Parameters

- **data_flow** (*rafcon.core.data_flow.DataFlow*) – The data flow to be wrapped
- **parent** (*rafcon.gui.models.abstract_state.AbstractStateModel*) – The state model of the state element

- **meta** (`rafcon.utils.vividict.Vividict`) – The meta data of the state element model

property core_element

Return the core element represented by this model

Returns

core element of the model

Return type

`rafcon.core.state_elements.state_element.StateElement`

property data_flow**get_observable_data_flow()****model_changed(model, prop_name, info)**

This method notifies the parent state about changes made to the state element

prepare_destruction()

Prepares the model for destruction

Unregisters the model from observing itself.

set_observable_data_flow(value)**DataPortModel (in data_port)****class rafcon.gui.models.data_port.DataPortModel(data_port, parent, meta=None)**

Bases: `StateElementModel`

This model class manages a DataPort

Parameters

- **data_port** (`rafcon.core.data_port.DataPort`) – The input/output data port to be wrapped
- **parent** (`rafcon.gui.models.abstract_state.AbstractStateModel`) – The state model of the state element
- **meta** (`rafcon.utils.vividict.Vividict`) – The meta data of the state element model

property core_element

Return the core element represented by this model

Returns

core element of the model

Return type

`rafcon.core.state_elements.state_element.StateElement`

property data_port**get_observable_data_port()****model_changed(model, prop_name, info)**

This method notifies the parent state about changes made to the state element

prepare_destruction()

Prepares the model for destruction

Unregisters the model from observing itself.

set_observable_data_port(*value*)**ScopedVariableModel (in `scoped_variable`)****class rafcon.gui.models.scoped_variable.ScopedVariableModel(*scoped_variable*, *parent*, *meta=None*)**

Bases: *StateElementModel*

This model class manages a ScopedVariable

Parameters

- **scoped_variable** (*rafcon.core.scoped_variable.ScopedVariable*) – The scoped variable to be wrapped
- **parent** (*rafcon.gui.models.abstract_state.AbstractStateModel*) – The state model of the state element
- **meta** (*rafcon.utils.vividict.Vividict*) – The meta data of the state element model

property core_element

Return the core element represented by this model

Returns

core element of the model

Return type

rafcon.core.state_elements.state_element.StateElement

get_observable_scoped_variable()**model_changed(*model*, *prop_name*, *info*)**

This method notifies the parent state about changes made to the state element

prepare_destruction()

Prepares the model for destruction

Unregisters the model from observing itself.

property scoped_variable**set_observable_scoped_variable(*value*)****Selection (in `selection`)****class rafcon.gui.models.selection.Selection(*parent_signal=None*)**

Bases: *ModelMT*

This class contains the selected models of a state_machine

add(*args, **kwargs)

Check for changes in the selection

If the selection is changed by the decorated method, the internal core element lists are updated and a signal is emitted with the old and new selection as well as the name of the method that caused the change..

clear(*args, **kwargs)

Check for changes in the selection

If the selection is changed by the decorated method, the internal core element lists are updated and a signal is emitted with the old and new selection as well as the name of the method that caused the change..

property data_flows

Returns all selected data flows

Returns

Subset of the selection, only containing data flows

Return type

set

property focus

Returns the currently focused element

property focus_signal**get_all()**

Return a copy of the selection

Returns

Copy of the set of selected elements

Return type

set

get_observable_focus_signal()**get_observable_selection_changed_signal()****get_selected_elements_of_core_class(core_element_type)**

Returns all selected elements having the specified *core_element_type* as state element class

Returns

Subset of the selection, only containing elements having *core_element_type* as state element class

Return type

set

get_selected_state()

Return the first state within the selection

Returns

First state within the selection or None if there is none

Return type

AbstractStateModel

handle_new_selection(*args, **kwargs)

Check for changes in the selection

If the selection is changed by the decorated method, the internal core element lists are updated and a signal is emitted with the old and new selection as well as the name of the method that caused the change..

handle_prepared_selection_of_core_class_elements(*args, **kwargs)

Check for changes in the selection

If the selection is changed by the decorated method, the internal core element lists are updated and a signal is emitted with the old and new selection as well as the name of the method that caused the change..

property income

Alias for incomes()

property incomes

Returns all selected incomes

Returns

Subset of the selection, only containing incomes

Return type

set

property input_data_ports

Returns all selected input data ports

Returns

Subset of the selection, only containing input data ports

Return type

set

is_selected(model)

Checks whether the given model is selected

Parameters

model –

Returns

True if the model is within the selection, False else

Return type

bool

on_model_destruct(destructed_model, signal, info)

Deselect models that are being destroyed

property outcomes

Returns all selected outcomes

Returns

Subset of the selection, only containing outcomes

Return type

set

property output_data_ports

Returns all selected output data ports

Returns

Subset of the selection, only containing output data ports

Return type

set

remove(*args, **kwargs)

Check for changes in the selection

If the selection is changed by the decorated method, the internal core element lists are updated and a signal is emitted with the old and new selection as well as the name of the method that caused the change..

property scoped_variables

Returns all selected scoped variables

Returns

Subset of the selection, only containing scoped variables

Return type

set

property selection_changed_signal**set(*args, **kwargs)**

Check for changes in the selection

If the selection is changed by the decorated method, the internal core element lists are updated and a signal is emitted with the old and new selection as well as the name of the method that caused the change..

set_observable_focus_signal(value)**set_observable_selection_changed_signal(value)****property states**

Returns all selected states

Returns

Subset of the selection, only containing states

Return type

set

property transitions

Returns all selected transitions

Returns

Subset of the selection, only containing transitions

Return type

set

update_core_element_lists()

Maintains inner lists of selected elements with a specific core element class

rafccon.gui.models.selection.extend_selection()

Checks if the selection is to be extended

The selection is to be extended, if a special modifier key (typically <Ctrl>) is being pressed.

Returns

If to extend the selection

Return type

True

```
rafcon.gui.models.selection.reduce_to_parent_states(models)
```

Remove all state models that also have their parents in the list

The function filters the list of models, so that for no model in the list, one of its (grand-)parents is also in the list. E.g. if the input models consists of a hierarchy state with two of its child states, the resulting list only contains the hierarchy state.

Parameters

models (*set*) – The set of selected models

Returns

The reduced set of selected models

Return type

set

```
rafcon.gui.models.selection.updates_selection(update_selection)
```

Decorator indicating that the decorated method could change the selection

LogicalPortModel (in `logical_port`)

```
class rafcon.gui.models.logical_port.IncomeModel(income, parent, meta=None)
```

Bases: *LogicalPortModel*

This model class manages an Income

Parameters

- **income** (*rafcon.core.logical_port.Income*) – The income to be wrapped
- **parent** (*rafcon.gui.models.abstract_state.AbstractStateModel*) – The state model of the state element
- **meta** (*rafcon.utils.vividict.Vividict*) – The meta data of the state element model

property core_element

Return the core element represented by this model

Returns

core element of the model

Return type

rafcon.core.state_elements.state_element.StateElement

```
get_observable_income()
```

```
property income
```

```
model_changed(model, prop_name, info)
```

This method notifies the parent state about changes made to the state element

```
set_observable_income(value)
```

```
class rafcon.gui.models.logical_port.LogicalPortModel(parent, meta=None)
```

Bases: *StateElementModel*

This model class serves as base class for all logical ports

```
class rafcon.gui.models.logical_port.OutcomeModel(outcome, parent, meta=None)
```

Bases: *LogicalPortModel*

This model class manages an Outcome

Parameters

- **outcome** (*rafcon.core.logical_port.Outcome*) – The outcome to be wrapped
- **parent** (*rafcon.gui.models.abstract_state.AbstractStateModel*) – The state model of the state element
- **meta** (*rafcon.utils.vividict.Vividict*) – The meta data of the state element model

property core_element

Return the core element represented by this model

Returns

core element of the model

Return type

rafcon.core.state_elements.state_element.StateElement

get_observable_outcome()

model_changed(model, prop_name, info)

This method notifies the parent state about changes made to the state element

property outcome

prepare_destruction()

Prepares the model for destruction

Unregisters the model from observing itself.

set_observable_outcome(value)

StateMachineModel (in state_machine)

```
class rafcon.gui.models.state_machine.ComplexActionObserver(model)
```

Bases: *Observer*

This Observer observes the and structures the information of complex actions and separates those observations from the StateMachineModel to avoid to mix these with the root state observation of the state machine model.

- In ongoing_complex_actions dictionary all active actions and there properties are hold
- Only once at the end of an complex action the ongoing_complex_actions dictionary is empty and the nested_action_already_in list has elements in to secure accessibility of action properties in the pattern
- The nested_action_already_in list is cleaned after the ongoing_complex_actions dictionary was cleared observable

action_signal(model, prop_name, info)

property ongoing_complex_actions

state_action_signal(model, prop_name, info)

```
class rafcon.gui.models.state_machine.StateMachineModel(**kwargs)
```

Bases: `MetaModel`, `Hashable`

This model class manages a `rafcon.core.state_machine.StateMachine`

The model class is part of the MVC architecture. It holds the data to be shown (in this case a state machine).

Parameters

`state_machine (StateMachine)` – The state machine to be controlled and modified

property action_signal

`action_signal_triggered(model, prop_name, info)`

When the action was performed, we have to set the dirty flag, as the changes are unsaved

`change_root_state_type(model, prop_name, info)`

property core_element

`destroy()`

property destruction_signal

`get_observable_action_signal()`

`get_observable_destruction_signal()`

`get_observable_meta_signal()`

`get_observable_ongoing_complex_actions()`

`get_observable_root_state()`

`get_observable_sm_selection_changed_signal()`

`get_observable_state_action_signal()`

`get_observable_state_machine()`

`get_observable_state_meta_signal()`

`get_state_model_by_path(path)`

Returns the `StateModel` for the given `path`

Searches a `StateModel` in the state machine, who's path is given by `path`.

Parameters

`path (str)` – Path of the searched state

Returns

The state with that path

Return type

`StateModel`

Raises

`ValueError`, if path is invalid/not existing with this state machine

`load_meta_data(path=None, recursively=True)`

Load meta data of state machine model from the file system

The meta data of the state machine model is loaded from the file system and stored in the `meta` property of the model. Existing meta data is removed. Also the meta data of root state and children is loaded.

Parameters

path (*str*) – Optional path to the meta data file. If not given, the path will be derived from the state machine’s path on the filesystem

meta = None

meta_changed(model, prop_name, info)

When the meta was changed, we have to set the dirty flag, as the changes are unsaved

property meta_signal

property ongoing_complex_actions

prepare_destruction()

Prepares the model for destruction

Unregister itself as observer from the state machine and the root state

property root_state

root_state_assign(model, prop_name, info)

root_state_model_after_change(model, prop_name, info)

root_state_model_before_change(model, prop_name, info)

selection = None

set_observable_action_signal(value)

set_observable_destruction_signal(value)

set_observable_meta_signal(value)

set_observable_ongoing_complex_actions(value)

set_observable_root_state(value)

set_observable_sm_selection_changed_signal(value)

set_observable_state_action_signal(value)

set_observable_state_machine(value)

set_observable_state_meta_signal(value)

property sm_selection_changed_signal

property state_action_signal

property state_machine

state_machine_model_after_change(model, prop_name, info)

property state_meta_signal

store_meta_data(copy_path=None)

Save meta data of the state machine model to the file system

This method generates a dictionary of the meta data of the state machine and stores it on the filesystem.

Parameters

copy_path (*str*) – Optional, if the path is specified, it will be used instead of the file system path

suppress_new_root_state_model_one_time = False

update_hash(*obj_hash*)

Should be implemented by derived classes to update the hash with their data fields

Parameters

obj_hash – The hash object (see Python hashlib)

update_meta_data_hash(*obj_hash*)

Should be implemented by derived classes to update the hash with their meta data fields

Parameters

obj_hash – The hash object (see Python hashlib)

ModificationsHistoryModel (in modification_history)

The History-Class provides the observation functionalities to register and identify all core or gui (graphical) edit actions that are a actual change to the state machine. Those changes are stored as Action-Objects in the ModificationsHistory-Class.

The HistoryChanges-Class provides the functionalities to organize and access all actions of the edit process. Hereby the branching of the edit process is stored and should be accessible, too.

```
class rafcon.gui.models.modification_history.HistoryTreeElement(prev_id=None,
                                                               next_id=None)
```

Bases: *object*

property history_id

property next_id

property old_next_ids

prepare_destruction()

property prev_id

```
class rafcon.gui.models.modification_history.ModificationsHistory
```

Bases: *Observable*

The Class holds a all time history and a trail history. The trail history holds directly all modifications made since the last reset until the actual last active change and the undone modifications of this branch of modifications. So the all time history holds a list of all modifications ordered by time whereby the list elements are TreeElements that know respective previous action's list id and the possible next action list ids (multiple branches). Hereby a fast search from a actual active branch (trail history) to specific history_id (some branch) can be performed and all recovery steps collected. Additionally there will be implemented functionalities that never forget a single change that was insert for debugging reasons. - the pointer are pointing on the next undo ... so redo is pointer + 1 - all_actions is a type of a tree # prev_id, action, next_id, old_next_ids

property current_history_element

get_current_branch_history_ids()

get_element_for_history_id(*history_id*)

```
get_executed_history_ids()
get_history_path_from_current_to_target_history_id(target_history_id)
get_next_element(for_history_element=None)
get_previous_element(for_history_element=None)
go_to_history_element(target_history_id)
insert_action(**kwargs)
is_redo_possible()
is_undo_possible()
prepare_destruction()
redo(**kwargs)
reset(**kwargs)
undo(**kwargs)

class rafcon.gui.models.modification_history.ModificationsHistoryModel(state_machine_model)
Bases: ModelMT

action_signal_after_complex_action(model, prop_name, info)
property active_action
after_count()

assign_notification_root_state_after(model, prop_name, info)
This method is called, when any state, transition, data flow, etc. within the state machine modifications.
This then typically requires a redraw of the graphical editor, to display these modifications immediately.
:param model: The state machine model :param prop_name: The property that was changed :param info:
Information about the change

assign_notification_root_state_before(model, prop_name, info)

assign_notification_states_after(model, prop_name, info)
This method is called, when any state, transition, data flow, etc. within the state machine modifications.
This then typically requires a redraw of the graphical editor, to display these modifications immediately.
:param model: The state machine model :param prop_name: The property that was changed :param info:
Information about the change

assign_notification_states_before(model, prop_name, info)

before_count()

property change_count
finish_new_action(overview)
get_observable_change_count()
get_observable_modifications()
get_root_state_element_meta()
```

```

get_state_element_meta_from_internal_tmp_storage(state_path)
meta_changed_notify_after(changed_model, prop_name, info)
property modifications

prepare_destruction()
    Prepares the model for destruction
    Un-registers itself as observer from the state machine and the root state

re_initiate_meta_data()

redo()

reset_to_history_id(target_history_id)
    Recovers a specific target_history_id of the _full_history element by doing several undos and redos.

        Parameters
            target_history_id – the id of the list element which is to recover

        Returns

set_observable_change_count(value)
set_observable_modifications(value)
start_new_action(overview)
start_new_action_old(overview)
state_action_signal(model, prop_name, info)
state_machine_model = None
synchronized_redo()
synchronized_undo()

property tmp_meta_storage

undo()

update_internal_tmp_storage()

```

AutoBackupModel (in auto_backup)

```

class rafcon.gui.models.auto_backup.AutoBackupModel(state_machine_model)
Bases: ModelMT

```

Class provides auto backup functionality for a state-machine.

The Class AutoBackupModel requests a threading.Lock object named storage_lock in the StateMachineModel handed to it to avoid inconsistencies if storing in the middle of a modification by API or e.g. ModificationHistory. The auto-backup class can be initiated but be disabled by TIMED_TEMPORARY_STORAGE_ENABLED in the gui_config.yaml. There are two mode to run this class – with fix interval checks for backup or by dynamical auto backup by setting the flag ONLY_FIX_FORCED_TEMPORARY_STORAGE_INTERVAL in the gui_config.yaml. The forced interval FORCED_TEMPORARY_STORAGE_INTERVAL is used for the fix auto backup interval and as the forced time interval for the dynamic auto backup. The dynamic auto backup will backup additionally if the user was not doing any modifications till a time

horizon of TIMED_TEMPORARY_STORAGE_INTERVAL. The flag AUTO_RECOVERY_CHECK enables the check on not cleanly closed instances and state machines what only can be performed if the AUTO_RECOVERY_LOCK_ENABLED is set True to write respective lock files into the backup folders. If the lock file is not cleaned up the state machine and the RAFCON instance was not closed cleanly.

cancel_timed_thread()

change_in_state_machine_notification(model, prop_name, info)

check_for_auto_backup(force=False)

The method implements the checks for possible auto backup of the state-machine according duration till the last change together with the private method _check_for_dyn_timed_auto_backup.

If the only_fix_interval is True this function is called ones in the beginning and is called by a timed- threads in a fix interval.

Parameters

force – is a flag that force the temporary backup of the state-machine to the tmp-folder

Returns

check_lock_file()

clean_lock_file(final=False)

destroy()

perform_temp_storage()

prepare_destruction()

Prepares the model for destruction

Unregister itself as observer from the state machine and the root state

set_timed_thread(duration, func, *args)

update_last_backup_meta_data()

Update the auto backup meta data with internal recovery information

update_last_sm_origin_meta_data()

Update the auto backup meta data with information of the state machine origin

update_tmp_storage_path()

write_backup_meta_data()

Write the auto backup meta data into the current tmp-storage path

rafcon.gui.models.auto_backup.check_for_crashed_rafcon_instances()

rafcon.gui.models.auto_backup.check_path_for_correct_dirty_lock_file(sm_path, path)

rafcon.gui.models.auto_backup.find_dirty_lock_file_for_state_machine_path(sm_path)

rafcon.gui.models.auto_backup.generate_rafcon_instance_lock_file()

rafcon.gui.models.auto_backup.move_dirty_lock_file(dirty_lock_file, sm_path)

Move the dirt_lock file to the sm_path and thereby is not found by auto recovery of backup anymore

rafcon.gui.models.auto_backup.recover_state_machine_from_backup(sm_path, pid=None, full_path_dirty_lock=None)

rafcon.gui.models.auto_backup.remove_rafcon_instance_lock_file()

StateMachineManagerModel (in state_machine_manager)

GlobalVariableManagerModel (in global_variable_manager)

LibraryManagerModel (in library_manager)

StateMachineExecutionEngineModel (in state_machine_execution_engine)

MVC Views: rafcon.gui.views

Contents

- *MVC Views: rafcon.gui.views*
 - *Subpackages of rafcon.gui.views*
 - *GraphicalEditor (in graphical_editor_gaphas)*
 - *ExecutionHistoryTreeView (in execution_history)*
 - *GlobalVariableEditorView (in global_variable_editor)*
 - *LibraryTreeView (in library_tree)*
 - *LoggingConsoleView (in logging_console)*
 - *MainWindowView (in main_window)*
 - *MenuBarView (in menu_bar)*
 - *ModificationHistoryView (in modification_history)*
 - *StateMachineTreeView (in state_machine_tree)*
 - *StateMachinesEditorView (in state_machines_editor)*
 - *StatesEditorView (in states_editor)*
 - *ToolBarView (in tool_bar)*

Subpackages of rafcon.gui.views

MVC State Editor Views (rafcon.gui.views.state_editor)

Contents

- *MVC State Editor Views (rafcon.gui.views.state_editor)*
 - *StateEditorView (in state_editor)*
 - *LinkageOverviewView (in linkage_overview)*
 - *InputPortsListView (in input_port_list)*
 - *OutputPortsListView (in output_port_list)*
 - *ScopedVariablesListView (in scoped_variables_list)*

- *SourceEditorView* (in *source_editor*)
- *StateDataFlowsListView* (in *data_flows*)
- *DescriptionEditorView* (in *description_editor*)
- *StateOutcomesEditorView* (in *state_outcomes*)
- *StateOverviewView* (in *state_overview*)
- *StateTransitionsEditorView* (in *state_transitions*)

StateEditorView (in state_editor)

A module to view the all aspects of a respective state.

```
class rafcon.gui.views.state_editor.state_editor.StateEditorView
    Bases: View

    bring_tab_to_the_top(tab_label)
        Find tab with label tab_label in list of notebook's and set it to the current page.

    Parameters
        tab_label – String containing the label of the tab to be focused

    icons = {'Data Linkage': '\uf0c1;', 'Description': '\uf036;', 'Linkage Overview': '\uf160;', 'Logical Linkage': '\uf1e0;', 'Semantic Data': '\uf19c;', 'Source': '\uf121;'}

    insert_scoped_variables_tab()
    insert_source_tab()
    prepare_the_labels()
    remove_scoped_variables_tab()
    remove_source_tab()
    set_default_paned_positions()
```

LinkageOverviewView (in linkage_overview)

```
class rafcon.gui.views.state_editor.linkage_overview.LinkageOverviewDataView
    Bases: TreeView

class rafcon.gui.views.state_editor.linkage_overview.LinkageOverviewLogicView
    Bases: TreeView

    property treeView

class rafcon.gui.views.state_editor.linkage_overview.LinkageOverviewView
    Bases: View
```

InputPortsListView (in input_port_list)

```
class rafcon.gui.views.state_editor.input_port_list.InputPortsListView
    Bases: TreeView
```

OutputPortsListView (in output_port_list)

```
class rafcon.gui.views.state_editor.output_port_list.OutputPortsListView
    Bases: TreeView
```

ScopedVariablesListView (in scoped_variables_list)

```
class rafcon.gui.views.state_editor.scoped_variables_list.ScopedVariablesListView
    Bases: TreeView
```

SourceEditorView (in source_editor)

```
class rafcon.gui.views.state_editor.source_editor.SourceEditorView
    Bases: EditorView
```

```
property button_container_min_width
```

```
on_draw(widget, event)
```

```
on_text_view_event(*args)
```

```
pane_position_check()
```

Update right bar pane position if needed

Checks calculates if the cursor is still visible and updates the pane position if it is close to not be seen. In case of an un-docked right-bar this method does nothing. :return:

StateDataFlowsListView (in data_flows)

A module to view the data flow of a respective state (internal, external)

```
class rafcon.gui.views.state_editor.data_flows.StateDataFlowsEditorView
    Bases: View
```

```
class rafcon.gui.views.state_editor.data_flows.StateDataFlowsListView
    Bases: TreeView
```

DescriptionEditorView (in `description_editor`)

```
class rafcon.gui.views.state_editor.description_editor.DescriptionEditorView
    Bases: EditorView
```

StateOutcomesEditorView (in `state_outcomes`)

```
class rafcon.gui.views.state_editor.outcomes.StateOutcomesEditorView
    Bases: View
class rafcon.gui.views.state_editor.outcomes.StateOutcomesTreeView
    Bases: TreeView
```

StateOverviewView (in `state_overview`)

```
class rafcon.gui.views.state_editor.overview.StateOverviewView
    Bases: View
```

StateTransitionsEditorView (in `state_transitions`)

A module to view the logical connections (transitions) of a respective state (internal, external).

```
class rafcon.gui.views.state_editor.transitions.StateTransitionsEditorView
    Bases: View
class rafcon.gui.views.state_editor.transitions.StateTransitionsListView
    Bases: TreeView
```

MVC View Utils (`rafcon.gui.views.utils`)

Contents

- *MVC View Utils (`rafcon.gui.views.utils`)*
 - *AboutDialogView (in `about_dialog`)*
 - *EditorView (in `editor`)*
 - *SingleWidgetWindowView (in `single_widget_window`)*

AboutDialogView (in about_dialog)

```
class rafcon.gui.views.utils.about_dialog.AboutDialogView
Bases: AboutDialog
```

EditorView (in editor)

```
class rafcon.gui.views.utils.editor.EditorView(name='SOURCE EDITOR', language='idl',
                                               editor_style='SOURCE_EDITOR_STYLE',
                                               run_with_spacer=False)
```

Bases: View

apply_tag(name)

code_changed(source)

get_buffer()

get_cursor_position()

get_text()

new_buffer()

register()

scroll_to_cursor_onscreen()

set_cursor_position(line_number, line_offset)

set_enabled(on, text=None)

Set the default input or deactivated (disabled) style scheme

The method triggers the signal ‘changed’ by using set_text. Therefore, the method use the while_in_set_enabled flag to make activities of the method observable. If a method trigger this method and was triggered by a changed-signal this flag is supposed to avoid recursive calls.

Parameters

- **on** (`bool`) – enable flag.
- **text** (`str`) – optional text to insert.

Returns

set_text(text)

The method insert text into the text buffer of the text view and preserves the cursor location.

Parameters

text (`str`) – which is insert into the text buffer.

Returns

SingleWidgetWindowView (in single_widget_window)

```
class rafcon.gui.views.utils.single_widget_window.SingleWidgetWindowView(view_class,
                           width=500,
                           height=500,
                           title=None,
                           pos=None)
```

Bases: View

GraphicalEditor (in graphical_editor_gaphas)**ExecutionHistoryTreeView (in execution_history)**

```
class rafcon.gui.views.execution_history.ExecutionHistoryTreeView
```

Bases: View, TreeView

```
class rafcon.gui.views.execution_history.ExecutionHistoryView
```

Bases: View, ScrolledWindow

GlobalVariableEditorView (in global_variable_editor)

```
class rafcon.gui.views.global_variable_editor.GlobalVariableEditorView
```

Bases: View

LibraryTreeView (in library_tree)

```
class rafcon.gui.views.library_tree.LibraryTreeView
```

Bases: View, TreeView

LoggingConsoleView (in logging_console)

```
class rafcon.gui.views.logging_console.LoggingConsoleView
```

Bases: View

clean_buffer()

Delete all contents of the logging buffer.

clip_buffer()

Clip the logging buffer (if required) to avoid exceeding the maximum logging buffer lines.

static create_text_buffer()

property filtered_buffer

get_cursor_position()

get_line_number_next_to_cursor_with_string_within(s)

Find the closest occurrence of a string with respect to the cursor position in the text view

get_text_of_line(*line_number_or_iter*)

We are not going to modify ‘filtered_buffer’ here in any shape or form. But we need an exclusive lock here to insure the iters are still valid.

len()

print_message(*message, log_level*)

Add the logging requests to the event queue.

print_to_text_view(*text, text_buf, use_tag=None*)

Add the new rows to the logging buffer.

restore_cursor_position()

scroll_to_cursor_onscreen()

set_cursor_on_line_with_string(*s, line_offset=0*)

set_cursor_position(*line_number, line_offset*)

set_enables(*enables*)

static split_text(*text_to_split*)

Split text

Splits the debug text into its different parts: ‘Time’, ‘LogLevel + Module Name’, ‘Debug message’

Parameters

text_to_split – Text to split

Returns

List containing the content of text_to_split split up

store_cursor_position()

update_auto_scroll_mode()

Register or un-register signals for follow mode

MainWindowView (in main_window)

class rafcon.gui.views.main_window.MainWindowView

Bases: View

bring_tab_to_the_top(*tab_label*)

Find tab with label tab_label in list of notebooks and set it to the current page.

Parameters

tab_label – String containing the label of the tab to be focused

rotate_and_detach_tab_labels()

Rotates tab labels of a given notebook by 90 degrees and makes them detachable.

Parameters

notebook – GTK Notebook container, whose tab labels are to be rotated and made detachable

MenuBarView (in menu_bar)

```
class rafcon.gui.views.menu_bar.MenuBarView
    Bases: View

    buttons = {'about': '\uf0a3;', 'add': '\uf067;', 'backward_step': '\uf3e5;',
    'bake_state_machine': '\uf187;', 'copy': '\uf0c5;', 'cut': '\uf0c4;',
    'data_flow_mode': None, 'delete': '\uf1f8;', 'expert_view': '\uf06e;',
    'full_screen': None, 'group': '\uf247;', 'is_start_state': '\uf0c8;',
    'layout_state_machine': '\uf0d0;', 'menu_preferences': '\uf0ad;', 'new':
    '\uf15b;', 'only_run_selected': '\uf5d2;', 'open': '\uf07c;', 'open_recent':
    '\uf07c;', 'paste': '\uf0ea;', 'pause': '\uf04c;', 'quit': '\uf057;', 'redo':
    '\uf01e;', 'refresh_all': '\uf021;', 'refresh_libraries': '\uf021;',
    'run_selected': '\uf192;', 'run_to_selected': '\uf2f6;', 'save': '\uf0c7;',
    'save_as': '\uf0c7;', 'save_as_copy': '\uf0c7;', 'save_state_as': '\uf0c7;',
    'show_aborted_preempted': None, 'show_data_flows': None, 'show_data_values':
    None, 'show_transitions': None, 'start': '\uf04b;', 'start_from_selected':
    '\uf2f5;', 'step_into': '\uf019;', 'step_mode': '\uf54b;', 'step_out':
    '\uf093;', 'step_over': '\uf064;', 'stop': '\uf04d;', 'substitute_state':
    '\uf021;', 'undo': '\uf0e2;', 'ungroup': '\uf248;'}
    set_menu_item_accelerator(menu_item_name, accel_code, remove_old=False)
    set_menu_item_icon(menu_item_name, uni_code=None)
    set_menu_item_sensitive(menu_item_name, sensitive)
    sub_menus = ['submenu_file', 'submenu_edit', 'submenu_view', 'submenu_execution',
    'submenu_help']
```

ModificationHistoryView (in modification_history)

```
class rafcon.gui.views.modification_history.HistoryTreeView
    Bases: View, TreeView

class rafcon.gui.views.modification_history.ModificationHistoryView
    Bases: View, ScrolledWindow
```

StateMachineTreeView (in state_machine_tree)

```
class rafcon.gui.views.state_machine_tree.StateMachineTreeView
    Bases: View, TreeView
```

StateMachinesEditorView (in state_machines_editor)

```
class rafcon.gui.views.state_machines_editor.StateMachinesEditorView
    Bases: View
```

StatesEditorView (in states_editor)

```
class rafcon.gui.views.states_editor.StatesEditorView
    Bases: View
    button_released(widget, event=None)
```

ToolBarView (in tool_bar)

```
class rafcon.gui.views.tool_barToolBarView
    Bases: View
```

Helper functions: rafcon.gui.helpers

This package contains all GUI helper modules.

Every module covers actions for specific fields concerning labels, text formatting or actions on core objects that also have gui elements like there models (here the state and the state machine).

label

```
rafcon.gui.helpers.label.append_sub_menu_to_parent_menu(name, parent_menu, icon_code=None)
```

```
rafcon.gui.helpers.label.create_button_label(icon, font_size='10')
```

Create a button label with a chosen icon.

Parameters

- **icon** – The icon
- **font_size** – The size of the icon

Returns

The created label

```
rafcon.gui.helpers.label.create_check_menu_item(label_text='', is_active=False, callback=None,
                                                callback_args=(), is_sensitive=True,
                                                accel_code=None, accel_group=None)
```

```
rafcon.gui.helpers.label.create_label_widget_with_icon(icon, text, tooltip=None)
```

```
rafcon.gui.helpers.label.create_left_bar_window_title(upper_title, lower_title)
```

Create the title of the un-docked left-bar window based on the open tabs in the upper and lower notebooks.

Parameters

- **upper_title** – The title of the currently-opened tab in the upper notebook
- **lower_title** – The title of the currently-opened tab in the lower notebook

Returns

The un-docked left-bar window title as a String

```
rafcon.gui.helpers.label.create_menu_box_with_icon_and_label(label_text)
```

Creates a MenuItem box, which is a replacement for the former ImageMenuItem. The box contains, a **label**

for the icon and one for the text.

Parameters

label_text – The text, which is displayed for the text label

Returns

```
rafcon.gui.helpers.label.create_menu_item(label_text='', icon_code='&#xf0c5;', callback=None,  
callback_args=(), accel_code=None, accel_group=None)
```

```
rafcon.gui.helpers.label.create_tab_header_label(tab_name, icons)
```

Create the tab header labels for notebook tabs. If USE_ICONS_AS_TAB_LABELS is set to True in the gui_config, icons are used as headers. Otherwise, the titles of the tabs are rotated by 90 degrees.

Parameters

- **tab_name** – The label text of the tab, written in small letters and separated by underscores, e.g. states_free
- **icons** – A dict mapping each tab_name to its corresponding icon

Returns

The GTK Eventbox holding the tab label

```
rafcon.gui.helpers.label.create_widget_title(title, widget_name=None)
```

```
rafcon.gui.helpers.label.ellipsize_labels_recursively(widget, ellipsize=<enum  
PANGO_ELLIPSIZE_END of type  
Pango.EllipsizeMode>, width_chars=1)
```

```
rafcon.gui.helpers.label.get_label_of_menu_item_box(menu_item)
```

```
rafcon.gui.helpers.label.get_notebook_tab_title(notebook, page_num)
```

Helper function that gets a notebook's tab title given its page number

Parameters

- **notebook** – The GTK notebook
- **page_num** – The page number of the tab, for which the title is required

Returns

The title of the tab

```
rafcon.gui.helpers.label.get_widget_title(tab_label_text)
```

Transform Notebook tab label to title by replacing underscores with white spaces and capitalizing the first letter of each word.

Parameters

tab_label_text – The string of the tab label to be transformed

Returns

The transformed title as a string

`rafcon.gui.helpers.label.is_event_of_key_string(event, key_string)`

Condition check if key string represent the key value of handed event and whether the event is of right type

The function checks for constructed event tuple that are generated by the rafcon.gui.shortcut_manager.ShortcutManager.
:param tuple event: Event tuple generated by the ShortcutManager
:param str key_string: Key string parsed to a key value and for condition check

`rafcon.gui.helpers.label.react_to_event(view, widget, event)`

Checks whether the widget is supposed to react to passed event

The function is intended for callback methods registering to shortcut actions. As several widgets can register to the same shortcut, only the one having the focus should react to it.

Parameters

- **view** (*gkmc3.View*) – The view in which the widget is registered
- **widget** (*Gtk.Widget*) – The widget that subscribed to the shortcut action, should be the top widget of the view
- **event** – The event that caused the callback

Returns

Whether the widget is supposed to react to the event or not

Return type

`bool`

`rafcon.gui.helpers.label.set_button_children_size_request(widget)`

`rafcon.gui.helpers.label.set_icon_and_text_box_of_menu_item(menu_item, uni_code)`

`rafcon.gui.helpers.label.set_label_markup(label, text, is_icon=False, size=None, letter_spacing=None)`

`rafcon.gui.helpers.label.set_notebook_title(notebook, page_num, title_label)`

Set the title of a GTK notebook to one of its tab's titles

Parameters

- **notebook** – The GTK notebook
- **page_num** – The page number of a specific tab
- **title_label** – The GTK label holding the notebook's title

Returns

The new title of the notebook

`rafcon.gui.helpers.label.set_window_size_and_position(window, window_key)`

Adjust GTK Window's size, position and maximized state according to the corresponding values in the runtime_config file. The maximize method is triggered last to restore also the last stored size and position of the window. If the runtime_config does not exist, or the corresponding values are missing in the file, default values for the window size are used, and the mouse position is used to adjust the window's position.

Parameters

- **window** – The GTK Window to be adjusted
- **window_key** – The window's key stored in the runtime config file

state (helper)

```
rafcon.gui.helpers.state.add_state(container_state_m, state_type, add_position=None)
```

Add a state to a container state

Adds a state of type state_type to the given container_state

Parameters

- **container_state_m** (*rafcon.gui.models.container_state.ContainerState*) – A model of a container state to add the new state to
- **state_type** (*rafcon.core.enums.StateType*) – The type of state that should be added
- **add_position** ((*float*, *float*)) – The position, to add the state at, relative to the container_state_m in item coordinates.

Returns

True if successful, False else

```
rafcon.gui.helpers.state.change_state_type(state_m, target_class)
```

```
rafcon.gui.helpers.state.check_expected_future_model_list_is_empty(target_state_m, msg,
delete=True,
with_logger=None)
```

Checks if the expected future models list/set is empty

Return False if there are still elements in and also creates a warning message as feedback.

Parameters

- **target_state_m** (*StateModel*) – The state model which expected_future_models attribute should be checked
- **msg** (*str*) – Message for the logger if a model is still in.
- **delete** (*bool*) – Flag to delete respective model from list/set.
- **with_logger** – A optional logger to use in case of logging messages

Return type

bool

Returns

True if empty and False if still model in set/list

```
rafcon.gui.helpers.state.create_new_state_from_state_with_type(source_state, target_state_class)
```

The function duplicates/transforms a state to a new state type. If the source state type and the new state type both are ContainerStates the new state will have not transitions to force the user to explicitly re-order the logical flow according the paradigm of the new state type.

Parameters

- **source_state** – previous/original state that is to transform into a new state type (target_state_class)
- **target_state_class** – the final state class type

Returns

```
rafcon.gui.helpers.state.create_state_model_for_state(new_state, meta, state_element_models)
```

Create a new state model with the defined properties

A state model is created for a state of the type of new_state. All child models in state_element_models (model list for port, connections and states) are added to the new model.

Parameters

- **new_state** (`StateModel`) – The new state object with the correct type
- **meta** (`Vividict`) – Meta data for the state model
- **state_element_models** (`list`) – All state element and child state models of the original state model

Returns

New state model for new_state with all childs of state_element_models

```
rafcon.gui.helpers.state.extract_child_models_of_state(state_m, new_state_class)
```

Retrieve child models of state model

The function extracts the child state and state element models of the given state model into a dict. It only extracts those properties that are required for a state of type *new_state_class*. Transitions are always left out.

Parameters

- **state_m** – state model of which children are to be extracted from
- **new_state_class** – The type of the new class

Returns

```
rafcon.gui.helpers.state.group_states_and_scoped_variables(state_m_list, sv_m_list)
```

```
rafcon.gui.helpers.state.insert_state_as(target_state_m, state, as_template)
```

Add a state into a target state

In case the state to be insert is a LibraryState it can be chosen to be insert as template.

Parameters

- **target_state_m** (`rafcon.gui.models.container_state.ContainerStateModel`)
– State model of the target state
- **state** (`rafcon.core.states.State`) – State to be insert as template or not
- **as_template** (`bool`) – The flag determines if a handed state of type LibraryState is insert as template

Returns

```
rafcon.gui.helpers.state.negative_check_for_model_in_expected_future_models(target_state_m,
                                                                           model, msg,
                                                                           delete=True,
                                                                           with_logger=None)
```

Checks if the expected future models list/set includes still a specific model

Return False if the handed model is still in and also creates a warning message as feedback.

Parameters

- **target_state_m** (`StateModel`) – The state model which expected_future_models attribute should be checked
- **model** (`Model`) – Model to check for.

- **msg** (*str*) – Message for the logger if a model is still in.
- **delete** (*bool*) – Flag to delete respective model from list/set.
- **with_logger** – A optional logger to use in case of logging messages

Return type

bool

Returns

True if empty and False if still model in set/list

`rafcon.gui.helpers.state.prepare_state_m_for_insert_as(state_m_to_insert, previous_state_size)`

Prepares and scales the meta data to fit into actual size of the state.

`rafcon.gui.helpers.state.substitute_state(target_state_m, state_m_to_insert, as_template=False)`

Substitutes the target state

Both, the state to be replaced (the target state) and the state to be inserted (the new state) are passed via parameters. The new state adapts the size and position of the target state. State elements of the new state are resized but keep their proportion.

Parameters

- **target_state_m** (*rafcon.gui.models.container_state.AbstractStateModel*) – State Model of state to be substituted
- **state_m_to_insert** (*rafcon.gui.models.container_state.StateModel*) – State Model of state to be inserted

Returns

`rafcon.gui.helpers.state.substitute_state_as(target_state_m, state, as_template, keep_name=False)`

Substitute a target state with a handed state

The method generates a state model for the state to be inserted and use function substitute_state to finally substitute the state. In case the state to be inserted is a LibraryState it can be chosen to be inserted as template. It can be chosen that the inserted state keeps the name of the target state.

Parameters

- **target_state_m** (*rafcon.gui.models.state.AbstractStateModel*) – State model of the state to be substituted
- **state** (*rafcon.core.states.State*) – State to be inserted
- **as_template** (*bool*) – The flag determines if a handed state of type LibraryState is insert as template
- **keep_name** (*bool*) – The flag to keep the name of the target state

Returns

`rafcon.gui.helpers.state.toggle_show_content_flag_of_library_state_model(state_m)`

`rafcon.gui.helpers.state.ungroup_state(state_m)`

`rafcon.gui.helpers.state.update_models_recursively(state_m, expected=True)`

If a state model is reused the model depth maybe is to low. Therefore this method checks if all library state models are created with reliable depth

Parameters

- **expected** (*bool*) – Define newly generated library models as expected or triggers logger warnings if False

state_machine (helper)

This module covers functionality which is state machine model related, e.g. use selection, dialogs ,storage and that are basically menu bar functions. Further the it holds methods that are not StateModel based and more generic. Additional this module holds methods that employing the state machine manager. Maybe this changes in future.

```
rafcon.gui.helpers.state_machine.add_data_port_to_selected_states(data_port_type,
                                                                data_type=None,
                                                                selected_states=None)
```

```
rafcon.gui.helpers.state_machine.add_new_state(state_machine_m, state_type, target_position=None)
```

Triggered when shortcut keys for adding a new state are pressed, or Menu Bar “Edit, Add State” is clicked.

Adds a new state only if the parent state (selected state) is a container state, and if the graphical editor or the state machine tree are in focus.

Parameters

- **state_machine_m** – the state machine model to add the state to
- **state_type** – the state type of the state to be added
- **target_position** ((*float*, *float*)) – The position, to add the state at, relative to the graphical editor.

```
rafcon.gui.helpers.state_machine.add_outcome_to_selected_states(selected_states=None)
```

```
rafcon.gui.helpers.state_machine.add_scoped_variable_to_selected_states(data_type=None, se-
lected_states=None)
```

```
rafcon.gui.helpers.state_machine.add_state_by_drag_and_drop(state, data)
```

```
rafcon.gui.helpers.state_machine.auto_layout_state_machine()
```

```
rafcon.gui.helpers.state_machine.bake_selected_state_machine(path=None)
```

```
rafcon.gui.helpers.state_machine.change_background_color(state_model)
```

```
rafcon.gui.helpers.state_machine.change_state_type_with_error_handling_and_logger_messages(state_m,
tar-
get_class)
```

```
rafcon.gui.helpers.state_machine.delete_core_element_of_model(model, raise_exceptions=False,
                                                               recursive=True, destroy=True,
                                                               force=False)
```

Deletes respective core element of handed model of its state machine

If the model is one of state, data flow or transition, it is tried to delete that model together with its data from the corresponding state machine.

Parameters

- **model** – The model of respective core element to delete
- **raise_exceptions** (*bool*) – Whether to raise exceptions or only log errors in case of failures
- **destroy** (*bool*) – Access the destroy flag of the core remove methods

Returns

True if successful, False else

```
rafcon.gui.helpers.state_machine.delete_core_elements_of_models(models, raise_exceptions=True,  
                                                               recursive=True, destroy=True,  
                                                               force=False)
```

Deletes all respective core elements for the given models

Calls the `delete_core_element_of_model()` for all given models.

Parameters

- **models** – A single model or a list of models of respective core element to be deleted
- **raise_exceptions** (`bool`) – Whether to raise exceptions or log error messages in case of an error
- **destroy** (`bool`) – Access the destroy flag of the core remove methods

Returns

The number of models that were successfully deleted

```
rafcon.gui.helpers.state_machine.delete_selected_elements(state_machine_m)
```

```
rafcon.gui.helpers.state_machine.find_libraries_dependencies(library_path, new_library_path)
```

Find and resolve all dependencies of all libraries of a library directory

Parameters

- **library_path** (`str`) – the library path
- **new_library_path** (`str`) – the new library path

:rtype list(rafcon.core.state_machine.StateMachine) :return: library dependencies

```
rafcon.gui.helpers.state_machine.find_library_dependencies(library_os_path, library_path=None,  
                                                       library_name=None,  
                                                       new_library_path=None,  
                                                       new_library_name=None)
```

Find and resolve all dependencies of a library

Parameters

- **library_os_path** (`str`) – the library os path
- **library_path** (`str`) – the library path
- **library_name** (`str`) – the library name
- **new_library_path** (`str`) – the new library path
- **new_library_name** (`str`) – the new library name

:rtype list(rafcon.core.state_machine.StateMachine) :return: library dependencies

```
rafcon.gui.helpers.state_machine.find_library_root_dependencies(library_root_name,  
                                                               new_library_root_name)
```

Find and resolve all dependencies of all libraries of a library root

Parameters

- **library_root_name** (`str`) – the library root name
- **new_library_root_name** (`str`) – the new library root name

:rtype list(rafcon.core.state_machine.StateMachine) :return: library dependencies

```
rafcon.gui.helpers.state_machine.generate_linux_launch_files(target_path, config_path,
                                                               state_machine_path)

rafcon.gui.helpers.state_machine.get_root_state_description_of_sm_file_system_path(file_system_path)

rafcon.gui.helpers.state_machine.get_root_state_file_path(sm_file_system_path)

rafcon.gui.helpers.state_machine.get_root_state_name_of_sm_file_system_path(file_system_path)

rafcon.gui.helpers.state_machine.group_selected_states_and_scoped_variables()

rafcon.gui.helpers.state_machine.insert_state_into_selected_state(state, as_template=False)
```

Adds a State to the selected state

Parameters

- **state** – the state which is inserted
- **as_template** – If a state is a library state can be insert as template

Returns

boolean: success of the insertion

```
rafcon.gui.helpers.state_machine.is_selection_inside_of_library_state(state_machine_m=None,
                                                                     selected_elements=None)
```

Check if handed or selected elements are inside of library state

If no state machine model or selected_elements are handed the method is searching for the selected state machine and its selected elements. If selected_elements list is handed handed state machine model is ignored.

Parameters

- **state_machine_m** ([rafcon.gui.models.state_machine.StateMachineModel](#)) – Optional state machine model
- **selected_elements** ([list](#)) – Optional model list that is considered to be selected

Returns

True if elements inside of library state

```
rafcon.gui.helpers.state_machine.is_state_machine_stopped_to_proceed(selected_sm_id=None,
                                                                    root_window=None)
```

Check if state machine is stopped and in case request user by dialog how to proceed

The function checks if a specific state machine or by default all state machines have stopped or finished execution. If a state machine is still running the user is ask by dialog window if those should be stopped or not.

Parameters

- **selected_sm_id** – Specific state mine to check for
- **root_window** – Root window for dialog window

Returns

```
rafcon.gui.helpers.state_machine.new_state_machine(*args)

rafcon.gui.helpers.state_machine.open_library_state_separately()
```

`rafcon.gui.helpers.state_machine.open_state_machine(path=None, recent_opened_notification=False)`

Open a state machine from respective file system path

Parameters

- **path** (*str*) – file system path to the state machine
- **recent_opened_notification** (*bool*) – flags that indicates that this call also should update recently open

:rtype rafcon.core.state_machine.StateMachine :return: opened state machine

`rafcon.gui.helpers.state_machine.paste_into_selected_state(state_machine_m, cursor_position=None)`

Parameters

cursor_position ((*float*, *float*)) – the cursor position relative to the main window.

`rafcon.gui.helpers.state_machine.reduce_to_parent_states(models)`

`rafcon.gui.helpers.state_machine.refresh_after_relocate_and_rename(affected_libraries)`

Save all library dependencies, refresh the open libraries and the library tree view

Parameters

affected_libraries (*str*) – the affected libraries

`rafcon.gui.helpers.state_machine.refresh_all(force=False)`

Remove/close all libraries and state machines and reloads them freshly from the file system

Parameters

force (*bool*) – Force flag to avoid any checks

`rafcon.gui.helpers.state_machine.refresh_libraries()`

`rafcon.gui.helpers.state_machine.refresh_selected_state_machine()`

Reloads the selected state machine.

`rafcon.gui.helpers.state_machine.relocate_libraries(libraries_os_path, libraries_name, new_directory, logger=None)`

Relocate a library directory

Parameters

- **libraries_os_path** (*str*) – the libraries os path
- **libraries_name** (*str*) – the libraries name
- **new_directory** (*str*) – the new directory
- **logger** (*logger*) – the logger

`rafcon.gui.helpers.state_machine.relocate_library(library_os_path, library_path, library_name, new_directory, logger=None)`

Relocate a library

Parameters

- **library_os_path** (*str*) – the file system path of the library
- **library_path** (*str*) – the path of the library
- **library_name** (*str*) – the name of the library
- **new_directory** (*str*) – the new directory

- **logger** (*logger*) – the logger

```
rafcon.gui.helpers.state_machine.relocate_library_root(library_root_name, new_directory,  
logger=None)
```

Relocate a library root

Parameters

- **library_root_name** (*str*) – the library root name
- **new_directory** (*str*) – the new directory
- **logger** (*logger*) – the logger

```
rafcon.gui.helpers.state_machine.rename_library(library_os_path, new_library_os_path, library_path,  
library_name, new_library_name, logger=None)
```

Rename a library

Parameters

- **library_os_path** (*str*) – the library os path
- **new_library_os_path** (*str*) – the new library os path
- **library_path** (*str*) – the library path
- **library_name** (*str*) – the library name
- **new_library_name** (*str*) – the new library name
- **logger** (*logger*) – the logger

```
rafcon.gui.helpers.state_machine.rename_library_root(library_root_name, new_library_root_name,  
logger=None)
```

Rename a library root

Parameters

- **library_root_name** (*str*) – the library root name
- **new_library_root_name** (*str*) – the new library root name
- **logger** (*logger*) – the logger

```
rafcon.gui.helpers.state_machine.replace_all_libraries_by_template(state_model)
```

```
rafcon.gui.helpers.state_machine.save_all_libraries(target_path)
```

```
rafcon.gui.helpers.state_machine.save_library(library, path)
```

Save a library and its meta data to a path

Parameters

- **library** (*str*) – the library
- **path** (*str*) – the path

```
rafcon.gui.helpers.state_machine.save_library_config(target_path)
```

```
rafcon.gui.helpers.state_machine.save_library_dependencies(library_dependencies)
```

Save all library dependencies and their meta data to a path

Parameters

library_dependencies (*str*) – the library dependencies

`rafcon.gui.helpers.state_machine.save_open_libraries()`

Save all open libraries

`rafcon.gui.helpers.state_machine.save_selected_state_as()`

Save selected state as separate state machine

:return True if successfully stored, False if the storing process was canceled or stopped by condition fail :rtype bool: :raises exceptions.ValueError: If dialog response ids are out of bounds

`rafcon.gui.helpers.state_machine.save_state_machine(delete_old_state_machine=False,
recent_opened_notification=False,
as_copy=False, copy_path=None)`

Save selected state machine

The function checks if states of the state machine has not stored script data abd triggers dialog windows to take user input how to continue (ignoring or storing this script changes). If the state machine file_system_path is None function save_state_machine_as is used to collect respective path and to store the state machine. The delete flag will remove all data in existing state machine folder (if plugins or feature use non-standard RAFCON files this data will be removed)

Parameters

- **delete_old_state_machine** (`bool`) – Flag to delete existing state machine folder before storing current version
- **recent_opened_notification** (`bool`) – Flag to insert path of state machine into recent opened state machine paths
- **as_copy** (`bool`) – Store state machine as copy flag e.g. without assigning path to state_machine.file_system_path

Returns

True if the storing was successful, False if the storing process was canceled or stopped by condition fail

Rtype bool

`rafcon.gui.helpers.state_machine.save_state_machine_as(path=None,
recent_opened_notification=False,
as_copy=False)`

Store selected state machine to path

If there is no handed path the interface dialog “create folder” is used to collect one. The state machine finally is stored by the save_state_machine function.

Parameters

- **path** (`str`) – Path of state machine folder where selected state machine should be stored
- **recent_opened_notification** (`bool`) – Flag to insert path of state machine into recent opened state machine paths
- **as_copy** (`bool`) – Store state machine as copy flag e.g. without assigning path to state_machine.file_system_path

Returns

True if successfully stored, False if the storing process was canceled or stopped by condition fail

Rtype bool

```
rafcon.gui.helpers.state_machine.selected_state_toggle_is_start_state()
rafcon.gui.helpers.state_machine.substitute_selected_library_state_with_template(keep_name=True)
rafcon.gui.helpers.state_machine.substitute_selected_state(state, as_template=False,
keep_name=False)
```

Substitute the selected state with the handed state

Parameters

- **state** (`rafcon.core.states.state.State`) – A state of any functional type that derives from State
- **as_template** (`bool`) – The flag determines if a handed the state of type LibraryState is insert as template

Returns

```
rafcon.gui.helpers.state_machine.substitute_selected_state_and_use_choice_dialog()
rafcon.gui.helpers.state_machine.ungroup_selected_state()
```

text_formatting

```
rafcon.gui.helpers.text_formatting.format_default_folder_name(folder_name)
rafcon.gui.helpers.text_formatting.format_folder_name_human_readable(folder_name)
rafcon.gui.helpers.text_formatting.limit_string(text, max_length, separator='\\x2026;')
```

Gaphas extensions for RAFCON: `rafcon.gui.mygaphas`

Gaphas is a Python library for state-machine editors using Canvas: <https://github.com/gaphor/gaphas>

It is used here as library for the graphical editor as replacement for OpenGL. This modules contains all extensions necessary for RAFCON.

Contents

- *Gaphas extensions for RAFCON: rafcon.gui.mygaphas*
 - *Views*
 - * *Perpendicular Line*
 - * *ConnectionViews*
 - * *PortViews*
 - * *StateView and NameView*
 - *Utility functions*
 - * *Enumerations*
 - * *Helper methods for drawing operations*
 - * *General helper methods*

- *aspect*
- *canvas*
- *connector*
- *constraint*
- *guide*
- *painter*
- *segment*
- *tools*
- *view*

Views

Views are derived from `gaphas.item` and are the visual representations for core elements/models.

Perpendicular Line

The base class for the `rafcon.gui.mygaphas.items.connection.ConnectionView`.

```
class rafcon.gui.mygaphas.items.line.PerpLine(hierarchy_level)
```

Bases: Line

```
add_perp_waypoint(pos=(0, 0), begin=True)
```

```
add_waypoint(pos)
```

```
draw(context)
```

Draw the line itself. See Item.draw(context).

```
draw_head(context, port)
```

Default head drawer: move cursor to the first handle.

```
draw_tail(context, port)
```

Default tail drawer: draw line to the last handle.

```
end_handles(include_waypoints=False)
```

```
from_handle()
```

```
property from_port
```

```
get_parent_state_v()
```

```
static is_in_port(port)
```

```
static is_out_port(port)
```

```
property name
```

```
property parent
```

point(*pos*)

```
>>> a = Line()
>>> a.handles()[1].pos = 25, 5
>>> a._handles.append(a._create_handle((30, 30)))
>>> a.point((-1, 0))
1.0
>>> f'{a.point((5, 4)):.3f}'
'2.942'
>>> f'{a.point((29, 29)):.3f}'
'0.784'
```

remove()

remove_all_waypoints()

reset_from_port()

reset_to_port()

to_handle()

property to_port

property view

property waypoints

ConnectionViews

class rafcon.gui.mygaphas.items.connection.ConnectionPlaceholderView(*hierarchy_level*)

Bases: *ConnectionView*

class rafcon.gui.mygaphas.items.connection.ConnectionView(*hierarchy_level*)

Bases: *PerpLine*

apply_meta_data()

remove()

remove_connection_from_ports()

reset_port_for_handle(*handle*)

set_port_for_handle(*port, handle*)

class rafcon.gui.mygaphas.items.connection.DataFlowPlaceholderView(*hierarchy_level*)

Bases: *ConnectionPlaceholderView*

class rafcon.gui.mygaphas.items.connection.DataFlowView(*data_flow_m, hierarchy_level*)

Bases: *ConnectionView*

draw(*context*)

Draw the line itself. See Item.draw(context).

property model

```
property show_connection

class rafcon.gui.mygaphas.items.connection.TransitionPlaceholderView(hierarchy_level)
    Bases: ConnectionPlaceholderView

class rafcon.gui.mygaphas.items.connection.TransitionView(transition_m, hierarchy_level)
    Bases: ConnectionView

    draw(context)
        Draw the line itself. See Item.draw(context).

    property model

    property show_connection
```

PortViews

Each port (income, outcome, data ports) are represented as a view element.

```
class rafcon.gui.mygaphas.items.ports.DataPortView(in_port, parent, port_m, side)
    Bases: PortView

    draw(context, state)

    has_label()

    property model

    property name

    property port_id

class rafcon.gui.mygaphas.items.ports.IncomeView(income_m, parent)
    Bases: LogicPortView

    draw(context, state, highlight=False)

    property model

class rafcon.gui.mygaphas.items.ports.InputPortView(parent, port_m)
    Bases: DataPortView

    draw(context, state)

class rafcon.gui.mygaphas.items.ports.LogicPortView(in_port, name=None, parent=None,
                                                       side=SnappedSide.RIGHT)
    Bases: PortView

    Base class for ports connecting transitions

    A logic port is either a income our an outcome.

    draw(context, state, highlight)

class rafcon.gui.mygaphas.items.ports.OutcomeView(outcome_m, parent)
    Bases: LogicPortView

    draw(context, state, highlight=False)
```

```

has_label()
property model
property name
property outcome_id

class rafcon.gui.mygaphas.items.ports.OutputPortView(parent, port_m)
    Bases: DataPortView
    draw(context, state)

class rafcon.gui.mygaphas.items.ports.PortView(in_port, name=None, parent=None,
                                               side=SnappedSide.RIGHT)
    Bases: object
    add_connected_handle(handle, connection_view, moving=False)
    property connected
    property connected_connections
    property connected_incoming
    property connected_outgoing
    draw(context, state)
    draw_name(context, transparency, value)
    draw_port(context, fill_color, transparency, value=None)
    get_port_area(view)
        Calculates the drawing area affected by the (hovered) port
    property handle_pos
    handles()
    has_label()
    has_outgoing_connection()
    is_selected()
    property name
    property parent
    property port_side_size
    property port_size
    property pos
    remove_connected_handle(handle)
    property side

```

```
tmp_connect(handle, connection_view)
tmp_disconnect()

property view

class rafcon.gui.mygaphas.items.ports.ScopedVariablePortView(parent, scoped_variable_m)
    Bases: PortView
    draw(context, state)
    draw_name(context, transparency, only_calculate_size=False)
        Draws the name of the port
        Offers the option to only calculate the size of the name.

        Parameters
            • context – The context to draw on
            • transparency – The transparency of the text
            • only_calculate_size – Whether to only calculate the size

        Returns
            Size of the name

        Return type
            float, float

    property model
    property name
    property port_id
    property port_size
```

StateView and NameView

Each `rafcon.gui.mygaphas.items.state.StateView` holds a child item `rafcon.gui.mygaphas.items.state.NameView`, as the name of a state can be resized and repositioned.

```
class rafcon.gui.mygaphas.items.state.NameView(name, size)
    Bases: Element
    apply_meta_data()
    draw(context)
        Render the item to a canvas view. Context contains the following attributes:
            • cairo: the Cairo Context use this one to draw
            • view: the view that is to be rendered to
            • selected, focused, hovered, dropzone: view state of items (True/False)
            • draw_all: a request to draw everything, for bounding box calculations

    property model
```

```

property name
property parent
property position
remove()
property transparency
update_minimum_size()
property view

class rafcon.gui.mygaphas.items.state.StateView(state_m, size, background_color, hierarchy_level)
  Bases: Element

  A State has 4 handles (for a start): NW +—+ NE SW +—+ SE

  add_income(income_m)
  add_input_port(port_m)
  static add_keep_rect_within_constraint(canvas, parent, child)
  add_outcome(outcome_m)
  add_output_port(port_m)
  add_rect_constraint_for_port(port)
  add_scoped_variable(scoped_variable_m)
  apply_meta_data(recursive=False)
  property border_width
  child_state_views()
  connect_connection_to_port(connection_v, port, as_target=True)
  connect_to_income(connection_v, handle)
  connect_to_input_port(port_id, connection_v, handle)
  connect_to_outcome(outcome_id, connection_v, handle)
  connect_to_output_port(port_id, connection_v, handle)
  connect_to_scoped_variable_port(scoped_variable_id, connection_v, handle)
  property corner_handles
  draw(context)

  Render the item to a canvas view. Context contains the following attributes:
    • cairo: the Cairo Context use this one to draw
    • view: the view that is to be rendered to
    • selected, focused, hovered, dropzone: view state of items (True/False)
    • draw_all: a request to draw everything, for bounding box calculations

```

```
get_all_ports()
get_logic_ports()
get_port_for_handle(handle)
static get_state_drawing_area(state)
get_transitions()
property hovered
property income
input_port(port_id)
property inputs
property model
property moving
property name_view
outcome_port(outcome_id)
property outcomes
output_port(port_id)
property outputs
property parent
property position
remove()
    Remove recursively all children and then the StateView itself
remove_income()
remove_input_port(input_port_v)
remove_keep_rect_within_constraint_from_parent()
remove_outcome(outcome_v)
remove_output_port(output_port_v)
remove_scoped_variable(scoped_variable_port_v)
resize_all_children(old_size, paste=False)
scoped_variable(scoped_variable_id)
property scoped_variables
property selected
set_enable_flag_keep_rect_within_constraints(enable)
    Enable/disables the KeepRectangleWithinConstraint for child states
```

setup_canvas()

Called when the canvas is set for the item. This method can be used to create constraints.

show_content(*with_content=False*)

Checks if the state is a library with the *show_content* flag set

Parameters

with_content – If this parameter is *True*, the method return only True if the library represents a ContainerState

Returns

Whether the content of a library state is shown

property show_data_port_label**property transparency**

Calculates the transparency for the state

Returns

State transparency

Return type

float

update_minimum_size()**update_minimum_size_of_children()****property view**

Utility functions

Enumerations

rafcon.gui.mygaphas.utils.enums.Direction

alias of DIRECTION

class rafcon.gui.mygaphas.utils.enums.SnappedSide(*value*)

Bases: [Enum](#)

An enumeration.

BOTTOM = 4

LEFT = 1

RIGHT = 3

TOP = 2

next()

opposite()

prev()

Helper methods for drawing operations

```
rafcon.gui.mygaphas.utils.gap_draw_helper.draw_label_path(context, width, height, arrow_height,  
distance_to_port, port_offset)
```

Draws the path for an upright label

Parameters

- **context** – The Cairo context
- **width** (*float*) – Width of the label
- **height** (*float*) – Height of the label
- **distance_to_port** (*float*) – Distance to the port related to the label
- **port_offset** (*float*) – Distance from the port center to its border

```
rafcon.gui.mygaphas.utils.gap_draw_helper.draw_port_label(context, port, transparency, fill,  
label_position,  
show_additional_value=False,  
additional_value=None,  
only_extent_calculations=False)
```

Draws a normal label indicating the port name.

Parameters

- **context** – Draw Context
- **port** – The PortView
- **transparency** – Transparency of the text
- **fill** – Whether the label should be filled or not
- **label_position** – Side on which the label should be drawn
- **show_additional_value** – Whether to show an additional value (for data ports)
- **additional_value** – The additional value to be shown
- **only_extent_calculations** – Calculate only the extends and do not actually draw

```
rafcon.gui.mygaphas.utils.gap_draw_helper.get_col_rgba(color, transparency=None, opacity=None)
```

This class converts a Gdk.Color into its r, g, b parts and adds an alpha according to needs

If both transparency and opacity is None, alpha is set to 1 => opaque

Parameters

- **color** (*Gdk.Color*) – Color to extract r, g and b from
- **transparency** (*float* / *None*) – Value between 0 (opaque) and 1 (transparent) or None if opacity is to be used
- **opacity** (*float* / *None*) – Value between 0 (transparent) and 1 (opaque) or None if transparency is to be used

Returns

Red, Green, Blue and Alpha value (all between 0.0 - 1.0)

```
rafcon.gui.mygaphas.utils.gap_draw_helper.get_side_length_of_resize_handle(view, item)
```

Calculate the side length of a resize handle

Parameters

- **view** (`rafcon.gui.mygaphas.view.ExtendedGtkView`) – View
- **item** (`rafcon.gui.mygaphas.items.state.StateView`) – StateView

Returns

side length

Return type

float

`rafcon.gui.mygaphas.utils.gap_draw_helper.get_text_layout(cairo_context, text, size)``rafcon.gui.mygaphas.utils.gap_draw_helper.limit_value_string_length(value)`

This method limits the string representation of the value to MAX_VALUE_LABEL_TEXT_LENGTH + 3 characters.

Parameters**value** – Value to limit string representation**Returns**

String holding the value with a maximum length of MAX_VALUE_LABEL_TEXT_LENGTH + 3

General helper methods`rafcon.gui.mygaphas.utils.gap_helper.add_data_flow_to_state(from_port_m, to_port_m, add_data_port=False)`

Interface method between Gaphas and RAFCON core for adding data flows

The method checks the types of the given ports and their relation. From this the necessary parameters for the add_dat_flow method of the RAFCON core are determined. Also the parent state is derived from the ports.

Parameters

- **from_port_m** – Port model from which the data flow starts
- **to_port_m** – Port model to which the data flow goes to
- **add_data_port** – Boolean add the data port automatically

Returns

True if a data flow was added, False if an error occurred

`rafcon.gui.mygaphas.utils.gap_helper.add_transition_to_state(from_port_m, to_port_m)`

Interface method between Gaphas and RAFCON core for adding transitions

The method checks the types of the given ports (IncomeView or OutcomeView) and from this determines the necessary parameters for the add_transition method of the RAFCON core. Also the parent state is derived from the ports.

Parameters

- **from_port_m** – Port model from which the transition starts
- **to_port_m** – Port model to which the transition goes to

Returns

True if a transition was added, False if an error occurred

`rafcon.gui.mygaphas.utils.gap_helper.calc_rel_pos_to_parent(canvas, item, handle)`

This method calculates the relative position of the given item's handle to its parent

Parameters

- **canvas** – Canvas to find relative position in
- **item** – Item to find relative position to parent
- **handle** – Handle of item to find relative position to

Returns

Relative position (x, y)

`rafcon.gui.mygaphas.utils.gap_helper.create_new_connection(from_port_m, to_port_m)`

Checks the type of connection and tries to create it

If bot port are logical port,s a transition is created. If both ports are data ports (including scoped variable), then a data flow is added. An error log is created, when the types are not compatible.

Parameters

- **from_port_m** – The starting port model of the connection
- **to_port_m** – The end port model of the connection

Returns

True if a new connection was added

`rafcon.gui.mygaphas.utils.gap_helper.extend_extents(extents, factor=1.1)`

Extend a given bounding box

The bounding box (x1, y1, x2, y2) is centrally stretched by the given factor.

Parameters

- **extents** – The bound box extents
- **factor** – The factor for stretching

Returns

(x1, y1, x2, y2) of the extended bounding box

`rafcon.gui.mygaphas.utils.gap_helper.get_port_for_handle(handle, state)`

Looks for and returns the PortView to the given handle in the provided state

Parameters

- **handle** – Handle to look for port
- **state** – State containing handle and port

Returns

PortView for handle

`rafcon.gui.mygaphas.utils.gap_helper.get_relative_positions_of_waypoints(transition_v)`

This method takes the waypoints of a connection and returns all relative positions of these waypoints.

Parameters

transition_v – Transition view to extract all relative waypoint positions

Returns

List with all relative positions of the given transition

```
rafcon.gui.mygaphas.utils.gap_helper.update_meta_data_for_connection_waypoints(graphical_editor_view,  
connection_v,  
last_waypoint_list,  
publish=True)
```

This method updates the relative position metadata of the connections waypoints if they changed

Parameters

- **graphical_editor_view** – Graphical Editor the change occurred in
- **connection_v** – Transition/Data Flow that changed
- **last_waypoint_list** – List of waypoints before change
- **publish** (`bool`) – Whether to publish the changes using the meta signal

```
rafcon.gui.mygaphas.utils.gap_helper.update_meta_data_for_name_view(graphical_editor_view,  
name_v, publish=True)
```

This method updates the metadata of a name view.

Parameters

- **graphical_editor_view** – Graphical Editor view the change occurred in
- **name_v** – The name view which has been changed/moved
- **publish** – Whether to publish the changes of the meta data

```
rafcon.gui.mygaphas.utils.gap_helper.update_meta_data_for_port(graphical_editor_view, item,  
handle)
```

This method updates the metadata of the states ports if they changed.

Parameters

- **graphical_editor_view** – Graphical Editor the change occurred in
- **item** – State the port was moved in
- **handle** – Handle of moved port or None if all ports are to be updated

```
rafcon.gui.mygaphas.utils.gap_helper.update_meta_data_for_state_view(graphical_editor_view,  
state_v,  
affects_children=False,  
publish=True)
```

This method updates the metadata of a state view

Parameters

- **graphical_editor_view** – Graphical Editor view the change occurred in
- **state_v** – The state view which has been changed/moved
- **affects_children** – Whether the children of the state view have been resized or not
- **publish** – Whether to publish the changes of the metadata

aspect**canvas****class rafcon.gui.mygaphas.canvas.ItemProjection(*point*, *item_point*, *item_target*)**Bases: `object`

Project a point of item A into the coordinate system of item B.

The class os based on the implementation of gaphas.canvas.CanvasProjection.

property pos**class rafcon.gui.mygaphas.canvas.MyCanvas**Bases: `Canvas`**add(*item*, *parent=None*, *index=None*)**

Add an item to the canvas.

```
>>> c = Canvas()
>>> from gaphas import item
>>> i = item.Item()
>>> c.add(i)
>>> len(c._tree.nodes)
1
>>> i._canvas is c
True
```

add_port(*port_v*)**exchange_model(*old_model*, *new_model*)****get_first_view()**

Return first registered view object

get_parent(*item*)See `tree.Tree.get_parent()`.

```
>>> c = Canvas()
>>> from gaphas import item
>>> i = item.Item()
>>> c.add(i)
>>> ii = item.Item()
>>> c.add(ii, i)
>>> c.get_parent(i)
>>> c.get_parent(ii)
<gaphas.item.Item ...>
```

get_view_for_core_element(*core_element*, *parent_item=None*)

Searches and returns the View for the given core element

Parameters

- **core_element** – The core element of the searched view
- **parent_item** (`gaphas.item.Item`) – Restrict the search to this parent item

Returns

The view for the given core element or None if not found

get_view_for_model(*model*)

Searches and return the View for the given model

Parameters

model (*gtkmvc3.ModelMT*) – The model of the searched view

Returns

The view for the given model or None if not found

remove(*item*)

Remove item from the canvas.

```
>>> c = Canvas()
>>> from gaphas import item
>>> i = item.Item()
>>> c.add(i)
>>> c.remove(i)
>>> c._tree.nodes
[]
>>> i._canvas
```

remove_port(*port_v*)

resolve_constraint(*constraints*)

resolve_item_constraints(*item*)

update_root_items()

wait_for_update(*trigger_update=False*)

Update canvas and handle all events in the gtk queue

Parameters

trigger_update (*bool*) – Whether to call update_now() or not

connector

class rafcon.gui.mygaphas.connector.RectanglePointPort(*point, port_v=None*)

Bases: *PointPort*

glue(*pos*)

Calculates the distance between the given position and the port

Parameters

pos ((*float*, *float*)) – Distance to this position is calculated

Returns

Distance to port

Return type

float

property height

property width

constraint

```
class rafcon.gui.mygaphas.constraint.BorderWidthConstraint(north_west, south_east, border_width,
                                                               factor)
```

Bases: Constraint

solve_for(*var=None*)

Solve the constraint for a given variable. The variable itself is updated.

```
class rafcon.gui.mygaphas.constraint.KeepPointWithinConstraint(parent_nw, parent_se, child,
                                                               margin_method=None)
```

Bases: Constraint

Ensure that the point is within its parent

Attributes:

- parent_nw: NW coordinates of parent
- parent_se: SE coordinates of parent
- child: coordinates of child

solve_for(*var=None*)

Ensure that the children is within its parent

```
class rafcon.gui.mygaphas.constraint.KeepPortDistanceConstraint(anchor, point, port,
                                                               distance_func, incoming)
```

Bases: Constraint

solve_for(*var*)

Solve the constraint for a given variable. The variable itself is updated.

```
class rafcon.gui.mygaphas.constraint.KeepRectangleWithinConstraint(parent_nw, parent_se,
                                                               child_nw, child_se,
                                                               child=None,
                                                               margin_method=None)
```

Bases: Constraint

Ensure that the children is within its parent

Attributes:

- parent_nw: NW coordinates of parent
- parent_se: SE coordinates of parent
- child_nw: NW coordinates of child
- child_se: SE coordinates of child

solve_for(*var=None*)

Ensure that the children is within its parent

```
class rafcon.gui.mygaphas.constraint.KeepRelativePositionConstraint(anchor, point)
```

Bases: Constraint

solve_for(*var*)

Solve the constraint for a given variable. The variable itself is updated.

```
class rafcon.gui.mygaphas.constraint.PortRectConstraint(rect, point, port)
```

Bases: Constraint

Keeps ports on rectangular path along containing state :param rect: Rect (NWpos, SPos) specifying the path to bind port to :param point: Port position :param port: Port to bind

get_adjusted_border_positions()

Calculates the positions to limit the port movement to :return: Adjusted positions nw_x, nw_y, se_x, se_y

static limit_pos(p, se_pos, nw_pos)

Limits position p to stay inside containing state :param p: Position to limit :param se_pos: Bottom/Right boundary :param nw_pos: Top/Left boundary :return:

set_nearest_border()

Snaps the port to the correct side upon state size change

solve_for(var=None)

Solve the constraint for a given variable. The variable itself is updated.

update_distance_to_border()

update_port_side()

Updates the initial position of the port

The port side is ignored but calculated from the port position. Then the port position is limited to the four side lines of the state.

update_position(p)

guide

```
class rafcon.gui.mygaphas.guide.GuidedNameInMotion(item, view)
```

Bases: GuidedItemInMotion

move(pos)

Move the item. x and y are in view coordinates.

```
class rafcon.gui.mygaphas.guide.GuidedStateHandleInMotion(item, handle, view)
```

Bases: GuidedStateMixin, GuidedItemHandleInMotion

glue(pos, distance=None)

Glue to an item near a specific point.

Returns a ConnectionSink or None.

move(pos)

```
class rafcon.gui.mygaphas.guide.GuidedStateInMotion(item, view)
```

Bases: GuidedStateMixin, GuidedItemInMotion

move(pos)

Move the item. x and y are in view coordinates.

start_move(pos)

stop_move()

```
class rafcon.gui.mygaphas.guide.GuidedStateMixin
Bases: GuideMixin
    MARGIN = 5

    find_horizontal_guides(item_hedges, pdy, width, excluded_items)
    find_vertical_guides(item_vedges, pdx, height, excluded_items)
    get_excluded_items()
        Get a set of items excluded from guide calculation.
```

painter

segment

```
class rafcon.gui.mygaphas.segment.TransitionSegment(item, view)
```

Bases: LineSegment

This class is used to redefine the behavior of transitions and how new waypoints may be added. It checks if the waypoint that should be created is not between the perpendicular connectors to the ports.

property item

split(pos)

split_segment(segment, count=2)

Split one item segment into count equal pieces.

Two lists are returned

- list of created handles
- list of created ports

Parameters

segment

Segment number to split (starting from zero).

count

Amount of new segments to be created (minimum 2).

property view

tools

view

```
class rafcon.gui.mygaphas.view.ExtendedGtkView(graphical_editor_v, state_machine_m, *args)
```

Bases: GtkView, Observer

do_configure_event(event)

configure_event(self, event:Gdk.EventConfigure) -> bool

property focused_item

The item with focus (receives key events a.o.)

get_items_at_point(pos, selected=True, distance=0)

Return the items located at pos (x, y).

Parameters

- **selected** (`bool`) – if False returns first non-selected item
- **distance** (`float`) – Maximum distance to be considered as “at point” (in viewport pixel)

get_port_at_point(vpos, distance=10, exclude=None, exclude_port_fun=None)

Find item with port closest to specified position.

List of items to be ignored can be specified with *exclude* parameter.

Tuple is returned

- found item
- closest, connectable port
- closest point on found port (in view coordinates)

Parameters**vpos**

Position specified in view coordinates.

distance

Max distance from point to a port (default 10)

exclude

Set of items to ignore.

get_state_at_point(vpos, distance=10)**get_zoom_factor()**

Returns the current zoom factor of the view

The zoom factor can be read out from the view’s matrix. `_matrix[0]` should be equal `_matrix[3]`. Index 0 is for the zoom in x direction, index 3 for the y direction :return: Current zoom factor

property graphical_editor**handle_new_selection(items)**

Determines the selection

The selection is based on the previous selection, the currently pressed keys and the passes newly selected items

Parameters

items – The newly selected item(s)

hovered_handle = None**prepare_destruction()**

Get rid of circular references

```
queue_draw_item(*items)
Extends the base class method to allow Ports to be passed as item

Parameters
  items – Items that are to be redrawn

select_item(items)
Select an items. This adds items to the set of selected items.

property selected_items
Items selected by the view

unselect_all()
Clearing the selected_item also clears the focused_item.

unselect_item(item)
Unselect an item.
```

13.2.2 action

Action class for history

The Action-Class provides a general redo or undo functionality for any action, as long as the the class object was initialized with consistent arguments.

This general Action (one procedure for all possible edition) procedure is expansive and complex, therefore it is aimed to define specific _-Action-Classes for simple/specific edit actions.

```
class rafcon.gui.action.AbstractAction(parent_path, state_machine_model, overview=None)
```

Bases: `object`

```
action_type = None
after_overview = None
after_state_image = None
as_dict()
description()
get_state_image()
prepare_destruction()
redo()
set_after(overview)
undo()
```

```
class rafcon.gui.action.Action(parent_path, state_machine_model, overview)
```

Bases: `ModelMT, AbstractAction`

```
action_signal(model, prop_name, info)
static add_core_object_to_state(state, core_obj)
compare_models(previous_model, actual_model)
```

```

emit_undo_redo_signal(action_parent_m, affected_models, after)

get_state_changed()

get_state_image()

redo()
    General Redo, that takes all elements in the parent path state stored of the before action state machine status.
    :return:

static remove_core_object_from_state(state, core_obj)

undo()
    General Undo, that takes all elements in the parent path state stored of the after action state machine status.
    :return:

update_state(state, stored_state)

update_state_from_image(state, state_image)

class rafcon.gui.action.ActionDummy(parent_path=None, state_machine_model=None, overview=None)
    Bases: AbstractAction

class rafcon.gui.action.AddObjectAction(parent_path, state_machine_model, overview)
    Bases: Action

    The class handles all adding object action of 7 valid kinds (of In-OutputDataPort, ScopedVariable, DataFlow,
    Outcome, Transition and State)

    correct_reference_state(state, state_image_of_state, storage_path)

    possible_method_names = ['add_state', 'add_outcome', 'add_input_data_port',
    'add_output_data_port', 'add_transition', 'add_data_flow', 'add_scoped_variable']

    redo()

    Returns
        Redo of adding object action is simply done by adding the object again from the af-
        ter_state_image of the parent state.

    set_after(overview)

    undo()
        General Undo, that takes all elements in the parent path state stored of the after action state machine status.
        :return:

class rafcon.gui.action.CoreObjectIdentifier(core_obj_or_cls)
    Bases: object

    type_related_list_name_dict = {'DataFlow': 'data_flows', 'DeciderState': 'states',
    'InputDataPort': 'input_data_ports', 'Outcome': 'outcomes', 'OutputDataPort':
    'output_data_ports', 'ScopedVariable': 'scoped_variables', 'State': 'states',
    'Transition': 'transitions'}

class rafcon.gui.action.DataFlowAction(parent_path, state_machine_model, overview)
    Bases: StateElementAction

    static get_set_of_arguments(df)

```

```
possible_args = ['from_state', 'from_key', 'to_state', 'to_key']

possible_method_names = ['modify_origin', 'from_state', 'from_key', 'modify_target',
 'to_state', 'to_key']

redo()

undo()

update_data_flow_from_image(df, arguments)

class rafcon.gui.action.DataPortAction(parent_path, state_machine_model, overview)
    Bases: StateElementAction

    static get_set_of_arguments(dp)

    possible_args = ['name', 'default_value']

    possible_method_names = ['name', 'data_type', 'default_value', 'change_data_type']

    redo()

    undo()

    update_data_port_from_image(dp, arguments)

class rafcon.gui.action.MetaDataAction(parent_path, state_machine_model, overview)
    Bases: AbstractAction

    get_state_image()

    get_state_model_changed()

    redo()

    undo()

class rafcon.gui.action.OutcomeAction(parent_path, state_machine_model, overview)
    Bases: StateElementAction

    static get_set_of_arguments(oc)

    possible_args = ['name']

    possible_method_names = ['name']

    redo()

    undo()

    update_outcome_from_image(oc, arguments)

class rafcon.gui.action.RemoveObjectAction(parent_path, state_machine_model, overview)
    Bases: Action

    adjust_linkage()

    as_dict()

    correct_reference_state(state, state_image_of_state, storage_path)
```

```

diff_related_elements()
get_object_identifier()

possible_method_names = ['remove_state', 'remove_outcome', 'remove_input_data_port',
 'remove_output_data_port', 'remove_transition', 'remove_data_flow',
 'remove_scoped_variable']

redo()
    General Redo, that takes all elements in the parent path state stored of the before action state machine status.
    :return:

set_after(overview)
store_related_elements(linkage_dict)

undo()
    General Undo, that takes all elements in the parent path state stored of the after action state machine status.
    :return:

class rafcon.gui.action.ScopedVariableAction(parent_path, state_machine_model, overview)
    Bases: DataPortAction

class rafcon.gui.action.StateAction(parent_path, state_machine_model, overview)
    Bases: Action

as_dict()

static get_set_of_arguments(s)

possible_args = ['name', 'description', 'script_text', 'start_state_id',
 'library_name', 'library_path', 'version', 'state_copy',
 'input_data_port_runtime_values', 'output_data_port_runtime_values',
 'use_runtime_value_input_data_ports', 'use_runtime_value_output_data_ports',
 'set_input_runtime_value', 'set_output_runtime_value',
 'set_use_input_runtime_value', 'set_use_output_runtime_value', 'semantic_data']

possible_method_names = ['parent', 'name', 'description', 'script', 'script_text',
 'outcomes', 'input_data_ports', 'output_data_ports', 'states', 'scoped_variables',
 'data_flows', 'transitions', 'start_state_id', 'change_state_type',
 'add_input_data_port', 'remove_input_data_port', 'add_output_data_port',
 'remove_output_data_port', 'set_input_runtime_value', 'set_output_runtime_value',
 'set_use_input_runtime_value', 'set_use_output_runtime_value',
 'input_data_port_runtime_values', 'output_data_port_runtime_values',
 'use_runtime_value_input_data_ports', 'use_runtime_value_output_data_ports',
 'group_states', 'ungroup_state', 'substitute_state', 'paste', 'cut',
 'semantic_data', 'add_semantic_data', 'remove_semantic_data']

redo()
    General Redo, that takes all elements in the parent path state stored of the before action state machine status.
    :return:

set_after(overview)

substitute_dict = {'set_input_runtime_value': 'input_data_port_runtime_values',
 'set_output_runtime_value': 'output_data_port_runtime_values',
 'set_use_input_runtime_value': 'use_runtime_value_input_data_ports',
 'set_use_output_runtime_value': 'use_runtime_value_output_data_ports'}

```

```
undo()
    General Undo, that takes all elements in the parent path state stored of the after action state machine status.
    :return:

update_property_from_image(s, arguments)

class rafcon.gui.action.StateElementAction(parent_path, state_machine_model, overview)
    Bases: AbstractAction

    after_arguments = None

    as_dict()

    static get_set_of_arguments(elem)

    get_state_image()

    possible_args = []

    possible_method_names = []

    set_after(overview)

class rafcon.gui.action.StateImage(core_data, meta_data, state_path, semantic_data, file_system_path,
                                    script_text, children)
    Bases: tuple

    children
        Alias for field number 6

    core_data
        Alias for field number 0

    file_system_path
        Alias for field number 4

    meta_data
        Alias for field number 1

    script_text
        Alias for field number 5

    semantic_data
        Alias for field number 3

    state_path
        Alias for field number 2

class rafcon.gui.action.StateMachineAction(parent_path, state_machine_model, overview)
    Bases: Action, ModelMT

    The state machine action is currently only used for root state type changes and their undo/redo functionality.

    action_signal(model, prop_name, info)

    redo()
        General Redo, that takes all elements in the parent path state stored of the before action state machine status.
        :return:
```

undo()

General Undo, that takes all elements in the parent and :return:

update_root_state_from_image(state, state_image)

```
class rafcon.gui.action.TransitionAction(parent_path, state_machine_model, overview)
```

Bases: *StateElementAction*

static get_set_of_arguments(t)

```
possible_args = ['from_state', 'from_outcome', 'to_state', 'to_key']
```

```
possible_method_names = ['modify_origin', 'from_state', 'from_outcome',
'modify_target', 'to_state', 'to_outcome']
```

redo()**undo()****update_transition_from_image(t, arguments)**

```
rafcon.gui.action.check_state_model_for_is_start_state(state_model)
```

```
rafcon.gui.action.create_state_from_image(state_image)
```

```
rafcon.gui.action.create_state_image(state_m)
```

Generates a tuple that holds the state as yaml-strings and its meta data in a dictionary. The tuple consists of:
[0] json_str for state, [1] dict of child_state tuples, [2] dict of model_meta-data of self and elements [3] path of state in state machine [4] script_text [5] file system path [6] semantic data # states-meta - [state-, transitions-, data_flows-, outcomes-, inputs-, outputs-, scopes, states-meta]

Parameters

- **state** (*rafcon.core.states.state.State*) – The state that should be stored

Returns

- state_tuple tuple

```
rafcon.gui.action.get_state_element_meta(state_model, with_parent_linkage=True, with_verbose=False,
level=None)
```

```
rafcon.gui.action.insert_state_meta_data(meta_dict, state_model, with_verbose=False, level=None)
```

```
rafcon.gui.action.meta_dump_or_deepcopy(meta)
```

Function to observe meta data vivi-dict copy process and to debug it at one point

13.2.3 clipboard

```
class rafcon.gui.clipboard.Clipboard
```

Bases: *Observable*

A class to hold models and selection for later usage in cut/paste or copy/paste actions. In cut/paste action the selection stored is used while later paste. In a copy/paste actions

copy(selection, smart_selection_adaption=True)

Copy all selected items to the clipboard using smart selection adaptation by default

Parameters

- **selection** – the current selection

- **smart_selection_adaption** (`bool`) – flag to enable smart selection adaptation mode

Returns**cut**(*selection, smart_selection_adaption=False*)

Cuts all selected items and copy them to the clipboard using smart selection adaptation by default

Parameters

- **selection** – the current selection
- **smart_selection_adaption** (`bool`) – flag to enable smart selection adaptation mode

Returns**destroy()**

Destroys the clipboard by relieving all model references.

static destroy_all_models_in_dict(*target_dict*)

Method runs the prepare destruction method of models which are assumed in list or tuple as values within a dict

static do_selection_reduction_to_one_parent(*selection*)

Find and reduce selection to one parent state.

Parameters**selection** –**Returns**

state model which is parent of selection or None if root state

static do_smart_selection_adaption(*selection, parent_m*)

Reduce and extend transition and data flow element selection if already enclosed by selection

The smart selection adaptation checks and ignores directly data flows and transitions which are selected without selected related origin or targets elements. Additional the linkage (data flows and transitions) if those origins and targets are covered by the selected elements is added to the selection. Thereby the selection it self is manipulated to provide direct feedback to the user.

Parameters

- **selection** –
- **parent_m** –

Returns**get_action_arguments**(*target_state_m*)

Collect argument attributes for action signal

Use non empty list dict to create arguments for action signal msg and logger messages. The action parent model can be different then the target state model because logical and data port changes also may influence the linkage, see action-module (undo/redo).

Parameters**target_state_m** (`rafcon.gui.models.abstract_state.AbstractStateModel`) –
State model of target of action**Returns**

dict with lists of elements part of the action, action parent model

get_semantic_dictionary_list()

paste(*target_state_m*, *cursor_position=None*, *limited=None*, *convert=False*)

Paste objects to target state

The method checks whether the target state is a execution state or a container state and inserts respective elements and notifies the user if the parts can not be insert to the target state. - for ExecutionStates outcomes, input- and output-data ports can be inserted - for ContainerState additional states, scoped variables and data flows and/or transitions (if related) can be inserted

Related data flows and transitions are determined by origin and target keys and respective objects which has to be in the state machine selection, too. Thus, transitions or data flows without the related objects are not copied. :param *target_state_m*: state in which the copied/cut elements should be insert :param *cursor_position*: cursor position relative to the *target_state_m* as item coordinates, used to adapt meta data positioning of elements e.g states and via points. :return:

prepare_new_copy()

reset_clipboard_mapping_dicts()

Reset mapping dictionaries

set_semantic_dictionary_list(*semantic_data*)

`rafcon.gui.clipboard.singular_form(name)`

13.2.4 shortcut_manager

class rafcon.gui.shortcut_manager.ShortcutManager(*window*)

Bases: `object`

Handles shortcuts

Holds a mapping between shortcuts and action. Actions can be subscribed to. When a listed shortcut is triggered, all subscribers are notified.

add_callback_for_action(*action*, *callback*)

Adds a callback function to an action

The method checks whether both action and callback are valid. If so, the callback is added to the list of functions called when the action is triggered.

Parameters

- **action** (`str`) – An action like ‘add’, ‘copy’, ‘info’
- **callback** – A callback function, which is called when action is triggered. It retrieves the event as parameter

Returns

True is the parameters are valid and the callback is registered, False else

Return type

`bool`

destroy()

register_shortcuts()

remove_callback_for_action(*action*, *callback*)

Remove a callback for a specific action

This is mainly for cleanup purposes or a plugin that replaces a GUI widget.

Parameters

- **action** (*str*) – the action of which the callback is going to be removed
- **callback** – the callback to be removed

remove_callbacks_for_controller(*controller*)

remove_shortcuts()

trigger_action(*action, key_value, modifier_mask, **kwargs*)

Calls the appropriate callback function(s) for the given action

Parameters

- **action** (*str*) – The name of the action that was triggered
- **key_value** – The key value of the shortcut that caused the trigger
- **modifier_mask** – The modifier mask of the shortcut that caused the trigger
- **cursor_position** – The position of the cursor, relative to the main window.

Returns

Whether a callback was triggered

Return type

bool

update_shortcuts()

13.2.5 singleton (in rafcon.gui)

13.3 Utilities: utils

Contents

- *Utilities: utils*
 - *constants*
 - *decorators*
 - *dict_operations*
 - *execution_log*
 - *filesystem*
 - *geometry*
 - *hashable*
 - *i18n*
 - *installation*
 - *log*
 - *log_helpers*
 - *multi_event*

- *plugins (utils)*
- *profiler*
- *resources*
- *storage_utils*
- *timer*
- *type_helpers*
- *vividict*

13.3.1 constants

13.3.2 decorators

`rafcon.utils.decorators.avoid_parallel_execution(func)`

A decorator to avoid the parallel execution of a function.

If the function is currently called, the second call is just skipped.

Parameters

`func` – The function to decorate

Returns

13.3.3 dict_operations

`rafcon.utils.dict_operations.check_if_dict_contains_object_reference_in_values(object_to_check, dict_to_check)`

Method to check if an object is inside the values of a dict. A simple `object_to_check` in `dict_to_check.values()` does not work as it uses the `__eq__` function of the object and not the object reference.

Parameters

- `object_to_check` – The target object.
- `dict_to_check` – The dict to search in.

Returns

13.3.4 execution_log

`rafcon.utils.execution_log.log_to_DataFrame(execution_history_items, data_in_columns=[], data_out_columns=[], scoped_in_columns=[], scoped_out_columns=[], semantic_data_columns=[], throw_on_pickle_error=True)`

Returns all collapsed items in a table-like structure (pandas.DataFrame) with one row per executed state and a set of properties resp. columns (e.g. `state_name`, `outcome`, `run_id`) for this state. The data flow (`data_in/out`, `scoped_data_in/out`, `semantic_data`) is omitted from this table representation by default, as the different states have different data in-/out-port, `scoped_data`- ports and `semantic_data` defined. However, you can ask specific `data`-/`scoped_data`-ports and `semantic data` to be exported as table column, given they are primitive-valued, by

including the port / key names in the `{*}_selected-parameters`. These table-columns will obviously only be well-defined for states having this kind of port-name-/semantic-key and otherwise will contain a None-like value, indicating missing data.

The available data per execution item (row in the table) can be printed using `pandas.DataFrame.columns`.

```
rafcon.utils.execution_log.log_to_collapsed_structure(execution_history_items,
                                                      throw_on_pickle_error=True,
                                                      include_erroneous_data_ports=False,
                                                      full_next=False)
```

Collapsed structure means that all history items belonging to the same state execution are merged together into one object (e.g. CallItem and ReturnItem of an ExecutionState). This is based on the log structure in which all Items which belong together have the same run_id. The collapsed items hold input as well as output data (direct and scoped), and the outcome the state execution. :param dict execution_history_items: history items, in the simplest case directly the opened shelve log file :param bool throw_on_pickle_error: flag if an error is thrown if an object cannot be un-pickled :param bool include_erroneous_data_ports: flag if to include erroneous data ports :param bool full_next: flag to indicate if the next relationship has also to be created at the end of container states :return: - start_item, the StateMachineStartItem of the log file - next, a dict mapping run_id → run_id of the next executed state on the same hierarchy level - concurrent, a dict mapping run_id → []list of run_ids of the concurrent next executed states (if present) - hierarchy, a dict mapping run_id → run_id of the next executed state on the deeper hierarchy level (the start state within that HierarchyState) - items, a dict mapping run_id → collapsed representation of the execution of the state with that run_id :rtype: tuple

```
rafcon.utils.execution_log.log_to_ganttpplot(execution_history_items)
```

Example how to use the DataFrame representation

```
rafcon.utils.execution_log.log_to_raw_structure(execution_history_items)
```

Logging with raw structure. :param dict execution_history_items: history items, in the simplest case directly the opened shelve log file :return: - start_item, the StateMachineStartItem of the log file - previous, a dict mapping history_item_id → history_item_id of previous history item - next, a dict mapping history_item_id → history_item_id of the next history item (except if next item is a concurrent execution branch) - concurrent, a dict mapping history_item_id → []list of concurrent next history_item_ids (if present) - grouped, a dict mapping run_id → []list of history items with this run_id :rtype: tuple

13.3.5 filesystem

```
rafcon.utils.filesystem.clean_file_system_paths_from_not_existing_paths(file_system_paths)
```

Cleans list of paths from elements that do not exist

If a path is no more valid/existing, it is removed from the list.

Parameters

`file_system_paths` (`list[str]`) – list of file system paths to be checked for existing

```
rafcon.utils.filesystem.copy_file_or_folder(src, dst)
```

```
rafcon.utils.filesystem.create_path(path)
```

Creates a absolute path in the file system.

Parameters

`path` – The path to be created

```
rafcon.utils.filesystem.get_default_config_path()
```

```
rafcon.utils.filesystem.get_default_log_path()
```

```
rafcon.utils.filesystem.make_file_executable(filename)
rafcon.utils.filesystem.make_tarfile(output_filename, source_dir)
rafcon.utils.filesystem.read_file(file_path, filename=None)
```

Open file by path and optional filename

If no file name is given the path is interpreted as direct path to the file to be read. If there is no file at location the return value will be None to offer a option for case handling.

Parameters

- **file_path** (`str`) – Path string.
- **filename** (`str`) – File name of the file to be read.

Returns

None or str

```
rafcon.utils.filesystem.separate_folder_path_and_file_name(path)
```

```
rafcon.utils.filesystem.write_file(file_path, content, create_full_path=False)
```

13.3.6 geometry

```
rafcon.utils.geometry.cal_dist_between_2_coord_frame_aligned_boxes(box1_pos, box1_size,
                                                               box2_pos, box2_size)
```

Calculate Euclidean distance between two boxes those edges are parallel to the coordinate axis

The function decides condition based which corner to corner or edge to edge distance needs to be calculated.

Parameters

- **box1_pos** (`tuple`) – x and y position of box 1
- **box1_size** (`tuple`) – x and y size of box 1
- **box2_pos** (`tuple`) – x and y position of box 2
- **box2_size** (`tuple`) – x and y size of box 2

Returns

distance

```
rafcon.utils.geometry.deg2rad(degree)
```

Converts angle given in degrees into radian

Parameters

degree (`float`) – Angle to be converted

Returns

Angle in radian

Return type

`float`

```
rafcon.utils.geometry.dist(p1, p2)
```

Calculates the distance between two points

The function calculates the Euclidean distance between the two 2D points p1 and p2 :param p1: Tuple with x and y coordinate of the first point :param p2: Tuple with x and y coordinate of the second point :return: The Euclidean distance

`rafcon.utils.geometry.equal(t1, t2, digit=None)`

Compare two iterators and its elements on specific digit precision

The method assume that t1 and t2 are are list or tuple.

Parameters

- **t1** – First element to compare
- **t2** – Second element to compare
- **digit (int)** – Number of digits to compare

Return type

`bool`

Returns

True if equal and False if different

`rafcon.utils.geometry.point_left_of_line(point, line_start, line_end)`

Determines whether a point is left of a line

Parameters

- **point** – Point to be checked (tuple with x any y coordinate)
- **line_start** – Starting point of the line (tuple with x any y coordinate)
- **line_end** – End point of the line (tuple with x any y coordinate)

Returns

True if point is left of line, else False

13.3.7 hashable

`class rafcon.utils.hashable.Hashable`

Bases: `object`

`static get_object_hash_string(object_)`

`mutable_hash(obj_hash=None)`

Creates a hash with the (im)mutable data fields of the object

Example:

```
>>> my_obj = type("MyDerivedClass", (Hashable,), { "update_hash": lambda
    ↪self, h: h.update("RAFCON") })
>>> my_obj_hash = my_obj.mutable_hash()
>>> print('Hash: ' + my_obj_hash.hexdigest())
Hash: c8b2e32dc31c5282e4b9dbc6a9975b65bf59cd80a7cee66d195e320484df5c6
```

Parameters

`obj_hash` – The hash object (see Python hashlib)

Returns

The updated hash object

update_hash(*obj_hash*)

Should be implemented by derived classes to update the hash with their data fields

Parameters

obj_hash – The hash object (see Python hashlib)

static update_hash_from_dict(*obj_hash*, *object_*)

Updates an existing hash object with another Hashable, list, set, tuple, dict or stringifiable object

Parameters

- **obj_hash** – The hash object (see Python hashlib documentation)
- **object** – The value that should be added to the hash (can be another Hashable or a dictionary)

13.3.8 i18n

rafcon.utils.i18n.create_mo_files(*logger*)**rafcon.utils.i18n.setup_l10n(*logger=None*)**

Setup RAFCON for localization

Specify the directory, where the translation files (*.mo) can be found (rafcon/locale/) and set localization domain (“rafcon”).

Parameters

logger – which logger to use for printing (either logging.log or distutils.log)

13.3.9 installation

rafcon.utils.installation.install_fonts(*restart=False*)**rafcon.utils.installation.install_locally_required_files()****rafcon.utils.installation.installed_font_faces_for_font(*font_name*)****rafcon.utils.installation.started_in_virtualenv()****rafcon.utils.installation.started_without_installation()****rafcon.utils.installation.update_font_cache(*path*)**

13.3.10 log

exception rafcon.utils.log.RAFCONDeprecationWarning

Bases: `DeprecationWarning`

rafcon.utils.log.add_logging_level(*level_name*, *level_num*, *method_name=None*)

Add new logging level

Comprehensively adds a new logging level to the *logging* module and the currently configured logging class.

method_name becomes a convenience method for both *logging* itself and the class returned by *logging.getLoggerClass()* (usually just *logging.Logger*). If *method_name* is not specified, *level_name.lower()* is used.

Parameters

- **level_name** (*str*) – the level name
- **level_num** (*int*) – the level number/value

Raises

AttributeError – if the level name is already an attribute of the *logging* module or if the method name is already present

Example

```
>>> add_logging_level('TRACE', logging.DEBUG - 5)
>>> logging.getLogger(__name__).setLevel("TRACE")
>>> logging.getLogger(__name__).trace('that worked')
>>> logging.trace('so did this')
>>> logging.TRACE
5
```

`rafcn.utils.log.get_logger(name)`

Returns a logger for the given name

The function is basically a wrapper for `logging.getLogger` and only ensures that the namespace is within “rafcn.” and that the propagation is enabled.

Parameters

- name** (*str*) – The namespace of the new logger

Returns

Logger object with given namespace

Return type

`logging.Logger`

`class rafcn.utils.log.log_exceptions(logger=None)`

Bases: `object`

Decorator to catch all exceptions and log them

13.3.11 log_helpers

`class rafcn.utils.log_helpers.LoggingViewHandler`

Bases: `Handler`

A LoggingHandler for `Gtk.TextViews`

The `LoggingViewHandler` prints log messages in special `Gtk.TextView`’s that provide a `print_message` method. The views must register themselves to the handler. There can be multiple views registered for one handler.

`classmethod add_logging_view(name, text_view)`

`emit(record)`

Logs a new record

If a logging view is given, it is used to log the new record to. The code is partially copied from the `StreamHandler` class.

Parameters
record –

Returns

classmethod **remove_logging_view**(*name*)

class **rafcon.utils.log_helpers.NoHigherLevelFilter**(*level*)

Bases: **Filter**

Filter high log levels

A logging filter that filters out all logging records, whose level are smaller than the level specified in the constructor.

Variables
level – the highest level the filter will pass

filter(*record*)

Filter high log levels

Filters all records, whose logging level is smaller than the level specified in the constructor

Parameters
record –

Returns

13.3.12 multi_event

rafcon.utils.multi_event.create(**events*)

Creates a new multi_event

The multi_event listens to all events passed in the “events” parameter.

Parameters

events – a list of threading.Events

Returns

The multi_event

Return type

threading.Event

rafcon.utils.multi_event.or_clear(*self*)

A function to overwrite the default clear function of the threading.Event

Parameters

self – Reference to the event

rafcon.utils.multi_event.or_set(*self*)

A function to overwrite the default set function of threading.Events

Parameters

self – Reference to the event

rafcon.utils.multi_event.orify(*e, changed_callback*)

Add another event to the multi_event

Parameters

- **e** – the event to be added to the multi_event

- **changed_callback** – a method to call if the event status changes, this method has access to the multi_event

Returns

13.3.13 plugins (utils)

`rafcon.utils.plugins.load_plugins()`

Loads all plugins specified in the RAFCON_PLUGIN_PATH environment variable

`rafcon.utils.plugins.run_hook(hook_name, *args, **kwargs)`

Runs the passed hook on all registered plugins

The function checks, whether the hook is available in the plugin.

Parameters

- **hook_name** – Name of the hook, corresponds to the function name being called
- **args** – Arguments
- **kwargs** – Keyword arguments

`rafcon.utils.plugins.run_on_state_machine_execution_finished()`

Runs the on_state_machine_execution_finished methods of all registered plugins

`rafcon.utils.plugins.run_post_inits(setup_config)`

Runs the post_init methods of all registered plugins

Parameters

`setup_config` –

`rafcon.utils.plugins.run_pre_inits()`

Runs the pre_init methods of all registered plugins

13.3.14 profiler

`rafcon.utils.profiler.start(name)`

`rafcon.utils.profiler.stop(name, result_path='/tmp/rafcon-docs/2407', view=False)`

13.3.15 resources

`rafcon.utils.resources.get_data_file_path(*rel_path)`

Get a file installed as data_file (located in share folder)

`rafcon.utils.resources.get_repository_share_path()`

Get the share folder from the repository

If started from repository, the path to the share folder within the repository is returned, otherwise, *None*.

`rafcon.utils.resources.search_in_share_folders(*rel_path)`

Search all possible share folders for a given resource

13.3.16 storage_utils

```
rafcon.utils.storage_utils.get_current_time_string()
rafcon.utils.storage_utils.load_dict_from_yaml(path)
    Loads a dictionary from a yaml file :param path: the absolute path of the target yaml file :return:
rafcon.utils.storage_utils.load_objects_from_json(path, as_dict=False)
    Loads a dictionary from a json file.

Parameters
    path – The relative path of the json file.

Returns
    The dictionary specified in the json file

rafcon.utils.storage_utils.write_dict_to_json(dictionary, path, **kwargs)
    Write a dictionary to a json file. :param path: The relative path to save the dictionary to :param dictionary: The
    dictionary to get saved :param kwargs: optional additional parameters for dumper
```

13.3.17 timer

```
class rafcon.utils.timer.Timer(logger, name='', continues=True)
Bases: object
property duration
reset()
start()
stop(key)

rafcon.utils.timer.measure_time(func)
```

13.3.18 type_helpers

```
rafcon.utils.type_helpers.convert_string_to_type(string_value)
    Converts a string into a type or class

Parameters
    string_value – the string to be converted, e.g. “int”

Returns
    The type derived from string_value, e.g. int

rafcon.utils.type_helpers.convert_string_value_to_type_value(string_value, data_type)
    Helper function to convert a given string to a given data type

Parameters
    • string_value (str) – the string to convert
    • data_type (type) – the target data type

Returns
    the converted value
```

```
rafcon.utils.type_helpers.type_inherits_of_type(inheriting_type, base_type)
```

Checks whether *inheriting_type* inherits from *base_type*

Parameters

- **inheriting_type** (*str*) –
- **base_type** (*str*) –

Returns

True if *base_type* is base of *inheriting_type*

13.3.19 vividict

```
class rafcon.utils.vividict.Vividict(dictionary=None)
```

Bases: *dict*, YAMLObject, JSONObject

Extended dictionary for arbitrary nested access

A class which inherits from dict and can store an element for an arbitrary nested key. The single elements of the key do not have to exist beforehand.

```
classmethod from_dict(dictionary)
```

Creates a Vividict from a (possibly nested) python dict

Parameters

dictionary (*dict*) – Python dict to be converted to a Vividict

Returns

A Vividict with all key-values pairs from the given dict

Return type

Vividict

```
set_dict(new_dict)
```

Sets the dictionary of the Vividict

The method is able to handle nested dictionaries, by calling the method recursively.

Parameters

new_dict – The dict that will be added to the own dict

```
to_dict(*args, **kwargs)
```

Converts the Vividict to a common Python dict

Returns

A Python dict with all key-values pairs from this Vividict

Return type

dict

```
static vividict_to_dict(vividict, native_strings=False)
```

Helper method to create Python dicts from arbitrary Vividict objects

Parameters

vividict (*Vividict*) – A Vividict to be converted

Returns

A Python dict

Return type

dict

CHAPTER
FOURTEEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

b

barrier_concurrency_state, 129

c

concurrency_state, 131

config, 156

constants, 271

container_state, 132

custom_exceptions, 157

rafcon.core.config, 156

rafcon.core.constants, 158

rafcon.core.custom_exceptions, 157

rafcon.core.execution.base_execution_history,
112

rafcon.core.execution.consumer_manager, 114

rafcon.core.execution.consumers.abstract_execution_history_consumer,
119

rafcon.core.execution.consumers.file_system_consumer,
119

rafcon.core.execution.execution_engine, 115

rafcon.core.execution.execution_history_factory,
115

rafcon.core.execution.execution_history_items,
115

rafcon.core.execution.execution_status, 117

rafcon.core.execution.in_memory_execution_history,
117

rafcon.core.global_variable_manager, 158

rafcon.core.id_generator, 158

rafcon.core.interface, 159

rafcon.core.library_manager, 161

rafcon.core.script, 161

rafcon.core.singleton, 162

rafcon.core.state_elements.data_flow, 120

rafcon.core.state_elements.data_port, 121

rafcon.core.state_elements.logical_port, 123

rafcon.core.state_elements.scope, 124

rafcon.core.state_elements.state_element, 127

rafcon.core.state_elements.transition, 126

rafcon.core.state_machine, 162

rafcon.core.state_machine_manager, 164

rafcon.core.states.barrier_concurrency_state,
129

rafcon.core.states.concurrency_state, 131

rafcon.core.states.container_state, 132

rafcon.core.states.execution_state, 140

rafcon.core.states.hierarchy_state, 140

rafcon.core.states.library_state, 141

rafcon.core.states.preemptive_concurrency_state,
144

rafcon.core.states.state, 145

rafcon.core.storage.storage, 153

d

data_flow, 120

data_port, 121

decorators, 271

description_editor, 170

e

editor, 175

enums, 158

execution_engine, 115

execution_history, 117

execution_state, 140

extended_controller, 176

f

filesystem, 272

g

geometry, 273

global_variable_manager, 158

rafcon.gui.action, 262

rafcon.gui.clipboard, 267

rafcon.gui.controllers.execution_history, 178

rafcon.gui.controllers.global_variable_manager,
181

rafcon.gui.controllers.library_tree, 183

rafcon.gui.controllers.logging_console, 183

rafcon.gui.controllers.menu_bar, 185

rafcon.gui.controllers.modification_history,
189

rafcon.gui.controllers.state_editor.data_flows, 256
 168 rafcon.gui.mygaphas.connector, 257
rafcon.gui.controllers.state_editor.description, 258
 170 rafcon.gui.mygaphas.constraint, 258
 170 rafcon.gui.mygaphas.guide, 259
rafcon.gui.controllers.state_editor.linkage_overview, 245
 165 rafcon.gui.mygaphas.items.connection, 245
 165 rafcon.gui.mygaphas.items.line, 244
rafcon.gui.controllers.state_editor.outcomes, 246
 171 rafcon.gui.mygaphas.items.ports, 246
 171 rafcon.gui.mygaphas.items.state, 248
rafcon.gui.controllers.state_editor.overview, 260
 172 rafcon.gui.mygaphas.segment, 260
 172 rafcon.gui.mygaphas.utils.enums, 251
rafcon.gui.controllers.state_editor.scoped_variables, 252
 165 rafcon.gui.mygaphas.utils.gap_draw_helper,
 165 252
rafcon.gui.controllers.state_editor.source_editor, 253
 167 rafcon.gui.mygaphas.utils.gap_helper, 253
 167 rafcon.gui.mygaphas.view, 260
rafcon.gui.controllers.state_editor.state_editor, 269
 170 rafcon.gui.shortcut_manager, 269
 170 rafcon.gui.singleton, 270
rafcon.gui.controllers.state_editor.transitions, 228
 173 rafcon.gui.views.execution_history, 228
rafcon.gui.controllers.state_icons, 196
rafcon.gui.controllers.state_machine_tree, 228
 190 rafcon.gui.views.global_variable_editor, 228
rafcon.gui.controllers.states_editor, 192
rafcon.gui.controllers.tool_bar, 195
rafcon.gui.controllers.top_tool_bar, 195
rafcon.gui.controllers.undocked_window, 197
rafcon.gui.controllers.utils.editor, 175
rafcon.gui.controllers.utils.extended_controller, 226
 176 rafcon.gui.views.state_editor.input_port_list,
 176 225
rafcon.gui.controllers.utils.single_widget_window, 225
 178 rafcon.gui.views.state_editor.linkage_overview,
 178 224
rafcon.gui.helpers.label, 231
rafcon.gui.helpers.state, 234
rafcon.gui.helpers.state_machine, 237
rafcon.gui.helpers.text_formatting, 243
rafcon.gui.models.abstract_state, 198
rafcon.gui.models.auto_backup, 221
rafcon.gui.models.container_state, 204
rafcon.gui.models.data_flow, 209
rafcon.gui.models.data_port, 210
rafcon.gui.models.global_variable_manager, 225
 223 rafcon.gui.views.state_editor.outcomes, 226
rafcon.gui.models.library_manager, 223
rafcon.gui.models.library_state, 207
rafcon.gui.models.logical_port, 215
rafcon.gui.models.modification_history, 219
rafcon.gui.models.scoped_variable, 211
rafcon.gui.models.selection, 211
rafcon.gui.models.state, 202
rafcon.gui.models.state_element, 208
rafcon.gui.models.state_machine, 216
rafcon.gui.models.state_machine_execution_engine, 228
 223 rafcon.gui.views.state_editor.output_port_list,
 223 225
rafcon.gui.models.state_machine_manager, 223
rafcon.gui.models.transition, 209
 h hierarchy_state, 140

i
id_generator, 158
interface, 159

l
library_manager, 161
library_state, 141
library_tree, 183
linkage_overview, 165
log, 275

m
menu_bar, 185
modification_history, 219
multi_event, 277

o
outcome, 123

p
plugins, 278
preemptive_concurrency_state, 144

r
resources, 278

s
scoped_variable, 124
scoped_variable_list, 165
script, 161
single_widget_window, 178
singleton, 162
source_editor, 167
state, 145
state_data_flows, 168
state_editor, 170
state_element, 127
state_icons, 196
state_machine, 162
state_machine_manager, 164
state_machine_status, 117
state_machine_tree, 190
state_outcomes, 171
state_overview, 172
state_transitions, 173
states_editor, 192
storage, 153
storage_utils, 279

t
tool_bar, 195
top_tool_bar, 195
transition, 126

u
rafcon.utils.constants, 271
rafcon.utils.decorators, 271
rafcon.utils.dict_operations, 271
rafcon.utils.execution_log, 271
rafcon.utils.filesystem, 272
rafcon.utils.geometry, 273
rafcon.utils.hashable, 274
rafcon.utils.i18n, 275
rafcon.utils.installation, 275
rafcon.utils.log, 275
rafcon.utils.log_helpers, 276
rafcon.utils.multi_event, 277
rafcon.utils.plugins, 278
rafcon.utils.profiler, 278
rafcon.utils.resources, 278
rafcon.utils.storage_utils, 279
rafcon.utils.timer, 279
rafcon.utils.type_helpers, 279
rafcon.utils.vividict, 280
undocked_window, 197

v
vividict, 280

INDEX

A

AboutDialogView (class in *rafcon.gui.views.utils.about_dialog*), 227
AbstractAction (class in *rafcon.gui.action*), 262
AbstractExecutionHistoryConsumer (class in *rafcon.core.execution.consumers.abstract_execution_history.Consumer*), 119
AbstractStateModel (class in *rafcon.gui.models.abstract_state*), 198
acquire_modification_lock() (*rafcon.core.state_machine.StateMachine* method), 162
Action (class in *rafcon.gui.action*), 262
action_signal (*rafcon.gui.models.abstract_state.AbstractStateModel* property), 199
action_signal (*rafcon.gui.models.state_machine.StateMachineModel* property), 217
action_signal() (*rafcon.gui.action.Action* method), 262
action_signal() (*rafcon.gui.action.StateMachineAction* method), 266
action_signal() (*rafcon.gui.controllers.state_machine_tree.StateMachineTreeController* method), 190
action_signal() (*rafcon.gui.models.state_machine.ComplexActionObserver* method), 216
action_signal_after_complex_action() (*rafcon.core.modification_history.ModificationsHistoryModel* method), 220
action_signal_triggered() (*rafcon.gui.models.abstract_state.AbstractStateModel* method), 199
action_signal_triggered() (*rafcon.gui.models.state_machine.StateMachineModel* method), 217
action_type (*rafcon.gui.action.AbstractAction* attribute), 262
ActionDummy (class in *rafcon.gui.action*), 263
activate_state_tab() (*rafcon.gui.controllers.states_editor.StatesEditorController*)
method), 192
active (*rafcon.core.states.state.State* property), 145
active_action (*rafcon.gui.models.modification_history.ModificationsHistoryModel* property), 220
add() (*rafcon.gui.models.selection.Selection* method), 211
add() (*rafcon.gui.mygaphas.canvas.MyCanvas* method), 256
add_callback_for_action() (*rafcon.gui.shortcut_manager.ShortcutManager* method), 269
add_callback_to_shortcut_manager() (*rafcon.gui.controllers.menu_bar.MenuBarController* method), 185
add_clear_menu_item() (*rafcon.gui.controllers.logging_console.LoggingConsoleController* method), 183
add_connected_handle() (*rafcon.gui.mygaphas.items.ports.PortView* method), 247
add_controller() (*rafcon.gui.controllers.utils.extended_controller.ExtendedController* method), 176
add_core_object_to_state() (*rafcon.core.states.container_state.ContainerState* method), 132
add_data_flow() (*rafcon.core.states.container_state.ContainerState* method), 132
add_data_flow_to_state() (in module *rafcon.gui.mygaphas.utils.gap_helper*), 253
add_data_port_to_selected_states() (in module *rafcon.gui.helpers.state_machine*), 237
add_default_values_of_scoped_variables_to_scoped_data() (*rafcon.core.states.container_state.ContainerState* method), 133
add_history_item_to_queue() (*rafcon.core.execution.consumer_manager.ExecutionHistoryConsumer* method), 114
add_income() (*rafcon.gui.mygaphas.items.state.StateView* method), 249
add_input_data_port() (*rafcon.core.states.library_state.LibraryState*)

method), 142
add_input_data_port() (rafcon.core.states.state.State method), 145
add_input_data_to_scoped_data() (rafcon.core.states.container_state.ContainerState method), 133
add_input_port() (rafcon.gui.mygaphas.items.state.StateView method), 249
add_keep_rect_within_constraint() (rafcon.gui.mygaphas.items.state.StateView static method), 249
add_logging_level() (in module rafcon.utils.log), 275
add_logging_view() (rafcon.utils.log_helpers.LoggingViewHandler class method), 276
add_missing_model() (rafcon.gui.models.state.StateModel method), 202
add_new_state() (in module rafcon.gui.helpers.state_machine), 237
add_outcome() (rafcon.core.states.library_state.LibraryState method), 142
add_outcome() (rafcon.core.states.state.State method), 145
add_outcome() (rafcon.gui.mygaphas.items.state.StateView method), 249
add_outcome_to_selected_states() (in module rafcon.gui.helpers.state_machine), 237
add_output_data_port() (rafcon.core.states.library_state.LibraryState method), 142
add_output_data_port() (rafcon.core.states.state.State method), 145
add_output_port() (rafcon.gui.mygaphas.items.state.StateView method), 249
add_perp_waypoint() (rafcon.gui.mygaphas.items.line.PerpLine method), 244
add_port() (rafcon.gui.mygaphas.canvas.MyCanvas method), 256
add_rect_constraint_for_port() (rafcon.gui.mygaphas.items.state.StateView method), 249
add_scoped_variable() (rafcon.core.states.container_state.ContainerState method), 133
add_scoped_variable() (rafcon.gui.mygaphas.items.state.StateView method), 249
add_scoped_variable_to_selected_states() (in module rafcon.gui.helpers.state_machine), 237
add_semantic_data() (rafcon.core.states.state.State method), 145
add_state() (in module rafcon.gui.helpers.state), 234
add_state() (rafcon.core.states.barrier_concurrency_state.BarrierConcurrencyState method), 129
add_state() (rafcon.core.states.container_state.ContainerState method), 133
add_state_by_drag_and_drop() (in module rafcon.gui.helpers.state_machine), 237
add_state_editor() (rafcon.gui.controllers.states_editor.StatesEditorController method), 192
add_state_execution_output_to_scoped_data() (rafcon.core.states.container_state.ContainerState method), 133
add_transition() (rafcon.core.states.container_state.ContainerState method), 133
add_transition_to_state() (in module rafcon.gui.mygaphas.utils.gap_helper), 253
add_waypoint() (rafcon.gui.mygaphas.items.line.PerpLine method), 244
AddObjectAction (class in rafcon.gui.action), 263
adjust_linkage() (rafcon.gui.action.RemoveObjectAction method), 264
after_arguments (rafcon.gui.action.StateElementAction attribute), 266
after_count() (rafcon.gui.models.modification_history.ModificationsHistory method), 220
after_notification_of_parent_or_state() (rafcon.gui.controllers.state_editor.data_flows.StateDataFlowsListController method), 168
after_notification_of_parent_or_state_from_lists() (rafcon.gui.controllers.state_editor.data_flows.StateDataFlowsListController method), 168
after_notification_of_parent_or_state_from_lists() (rafcon.gui.controllers.state_editor.transitions.StateTransitionsListController method), 174
after_notification_of_script_text_was_changed() (rafcon.gui.controllers.utils.editor.EditorController method), 175
after_notification_state() (rafcon.gui.controllers.state_editor.transitions.StateTransitionsListController method), 174
after_overview (rafcon.gui.action.AbstractAction attribute), 262
after_state_image (rafcon.gui.action.AbstractAction attribute), 262
append_string_to_menu() (rafcon.gui.controllers.execution_history.ExecutionHistoryTreeController method), 179
append_sub_menu_to_parent_menu() (in module rafcon.gui.controllers.state_editor.TransitionEditorController), 179

con.gui.helpers.label), 231
apply_clicked() (raf- *con.gui.controllers.state_editor.source_editor.SourceEditorController*, 220
method), 167
apply_clicked() (raf- *con.gui.controllers.utils.editor.EditorController*
method), 175
apply_meta_data() (raf- *con.gui.mygaphas.items.connection.ConnectionView*
method), 245
apply_meta_data() (raf- *con.gui.mygaphas.items.state.NameView*
method), 248
apply_meta_data() (raf- *con.gui.mygaphas.items.state.StateView*
method), 249
apply_new_global_variable_name() (raf- *con.gui.controllers.global_variable_manager.GlobalVariableManagerController*, 271
method), 181
apply_new_global_variable_type() (raf- *con.gui.controllers.global_variable_manager.GlobalVariableManagerController*, 237
method), 181
apply_new_global_variable_value() (raf- *con.gui.controllers.global_variable_manager.GlobalVariableManagerController*, 222
method), 182
apply_new_outcome_name() (raf- *con.core.states.barrier_concurrency_state*,
con.gui.controllers.state_editor.outcomes.StateOutcomesListController
method), 171
apply_new_scoped_variable_default_value() (raf- *con.core.execution.base_execution_history*,
con.gui.controllers.state_editor.scoped_variable_list.ScopedVariableListController
method), 166
apply_new_scoped_variable_name() (raf- *con.gui.models.modification_history.ModificationsHistoryModel*
con.gui.controllers.state_editor.scoped_variable_list.ScopedVariableListController
method), 166
apply_new_scoped_variable_type() (raf- *con.gui.controllers.state_editor.data_flows.StateDataFlowsListController*
con.gui.controllers.state_editor.scoped_variable_list.ScopedVariableListController
method), 166
apply_tag() (rafcon.gui.views.utils.editor.EditorView
method), 227
as_dict() (rafcon.core.config.ObservableConfig
method), 156
as_dict() (rafcon.gui.action.AbstractAction
method), 262
as_dict() (rafcon.gui.action.RemoveObjectAction
method), 264
as_dict() (rafcon.gui.action.StateAction
method), 265
as_dict() (rafcon.gui.action.StateElementAction
method), 266
assign_notification_from_gvm() (raf- *con.gui.controllers.global_variable_manager.GlobalVariableManagerController*
method), 182
assign_notification_root_state_after() (raf- *con.gui.models.modification_history.ModificationsHistoryModel*
con.gui.views.state_editor.source_editor.SourceEditorView
method), 220
assign_notification_root_state_before() (raf- *con.gui.models.modification_history.ModificationsHistoryModel*
con.gui.controllers.state_machine.StateMachineTreeController, 220
method), 190
assign_notification_selection() (raf- *con.gui.controllers.state_machine.StateMachineTreeController*
method), 220
assign_notification_states_after() (raf- *con.gui.models.modification_history.ModificationsHistoryModel*
method), 220
assign_notification_states_before() (raf- *con.gui.models.modification_history.ModificationsHistoryModel*
method), 220
auto_layout_state_machine() (in module *rafcon.gui.helpers.state_machine*, 237
AutoBackupModel (class in *rafcon.gui.models.auto_backup*, 221
avoid_parallel_execution() (in module *rafcon.core.execution*, 271
B
take_selected_state_machine() (in module *rafcon.gui.helpers.state_machine*, 237
barrier_concurrency_state (rafcon.core.states.barrier_concurrency_state, 222
BarrierConcurrencyState (class in *rafcon.core.states.barrier_concurrency_state*,
con.core.states.barrier_concurrency_state, 222
BaseExecutionHistory (class in *rafcon.core.execution.base_execution_history*,
con.core.execution.base_execution_history, 222
before_count() (raf- *con.gui.controllers.state_editor.data_flows.StateDataFlowsListController*
con.gui.controllers.state_editor.scoped_variable_list.ScopedVariableListController
method), 166
before_notification_of_parent_or_state() (raf- *con.gui.controllers.state_editor.transitions.StateTransitionsListController*
con.gui.controllers.state_editor.transitions.StateTransitionsListController
method), 174
border_width (*rafcon.gui.mygaphas.items.state.StateView*
property), 249
BorderWidthConstraint (class in *rafcon.gui.mygaphas.constraint*, 258
BOTTOM (*rafcon.gui.mygaphas.utils.enums.SnappedSide*
attribute), 251
bring_tab_to_the_top() (raf- *con.gui.views.main_window.MainWindowView*
method), 229
bring_tab_to_the_top() (raf- *con.gui.views.state_editor.StateEditorView*
method), 224
button_container_min_width (raf- *rafcon.gui.views.state_editor.source_editor.SourceEditorView*
property), 225

button_released() (raf-
con.gui.views.states_editor.StatesEditorView
method), 231

buttons (rafcon.gui.views.menu_bar.MenuBarView attribute), 230

C

cal_dist_between_2_coord_frame_aligned_boxes() (in module rafcon.utils.geometry), 273

calc_rel_pos_to_parent() (in module raf-
con.gui.myaphas.utils.gap_helper), 253

call_action_callback() (raf-
con.gui.controllers.menu_bar.MenuBarController
method), 185

CallItem (class in raf-
con.core.execution.execution_history_items), 115

CallType (in module raf-
con.core.execution.execution_history_items), 115

cancel_clicked() (raf-
con.gui.controllers.utils.editor.EditorController
method), 175

cancel_timed_thread() (raf-
con.gui.models.auto_backup.AutoBackupModel
method), 222

change_background_color() (in module raf-
con.gui.helpers.state_machine), 237

change_count (rafcon.gui.models.modification_history.ModificationHistoryModel
property), 220

change_data_type() (raf-
con.core.state_elements.data_port.DataPort
method), 121

change_in_state_machine_notification() (raf-
con.gui.models.auto_backup.AutoBackupModel
method), 222

change_name() (rafcon.gui.controllers.state_editor.overview.StateOverviewController
method), 173

change_root_state_type() (raf-
con.core.state_machine.StateMachine method), 162

change_root_state_type() (raf-
con.gui.models.state_machine.StateMachineModel
method), 217

change_state_id() (raf-
con.core.states.container_state.ContainerState
method), 133

change_state_id() (rafcon.core.states.state.State
method), 145

change_state_type() (in module raf-
con.gui.helpers.state), 234

change_state_type() (raf-
con.core.states.container_state.ContainerState
method), 133

change_state_type_with_error_handling_and_logger_messages() (in module rafcon.gui.helpers.state_machine), 237

change_type() (rafcon.gui.controllers.state_editor.overview.StateOverviewController
method), 173

check_child_validity() (raf-
con.core.states.container_state.ContainerState
method), 133

check_child_validity() (raf-
con.core.states.state.State method), 145

check_data_flow_id() (raf-
con.core.states.container_state.ContainerState
method), 133

check_data_port_connection() (raf-
con.core.states.container_state.ContainerState
method), 134

check_default_value() (raf-
con.core.state_elements.data_port.DataPort
method), 122

check_edit_menu_items_status() (raf-
con.gui.controllers.menu_bar.MenuBarController
method), 186

check_expected_future_model_list_is_empty() (in module rafcon.gui.helpers.state), 234

check_for_auto_backup() (raf-
con.gui.models.auto_backup.AutoBackupModel
method), 222

check_for_crashed_rafcon_instances() (in mod-
ule rafcon.gui.models.auto_backup), 222

check_for_enter() (raf-
con.gui.controllers.state_editor.overview.StateOverviewController
method), 173

check_if_child_state_was_modified() (raf-
con.core.states.hierarchy_state.HierarchyState
method), 141

check_if_dict_contains_object_reference_in_values() (in module rafcon.gui.utils.dict_operations), 271

check_input_data_type() (raf-
con.core.states.state.State method), 146

check_lock_file() (raf-
con.gui.models.auto_backup.AutoBackupModel
method), 222

check_output_data_type() (raf-
con.core.states.state.State method), 146

check_path_for_correct_dirty_lock_file() (in module rafcon.gui.models.auto_backup), 222

check_state_model_for_is_start_state() (in module rafcon.gui.action), 267

check_transition_id() (raf-
con.core.states.container_state.ContainerState
method), 134

child_model_changed() (raf-
con.gui.models.abstract_state.AbstractStateModel
method), 199

child_state_views() (raf-
 con.gui.mygaphas.items.state.StateView
 method), 249

children (rafcon.gui.action.StateImage attribute), 266

clean_buffer() (raf-
 con.gui.views.logging_console.LoggingConsoleView
 method), 228

clean_file_system_paths_from_not_existing_paths() module, 156
 (in module rafcon.utils.filesystem), 272

clean_history() (raf-
 con.gui.controllers.execution_history.ExecutionHistoryTree
 method), 179

clean_lock_file() (raf-
 con.gui.models.auto_backup.AutoBackupModel
 method), 222

clean_path() (in module rafcon.core.storage.storage), 153

clean_path_element() (in module raf-
 con.core.storage.storage), 153

clean_path_from_deprecated_naming() (in module
 rafcon.core.storage.storage), 153

clear() (rafcon.gui.models.selection.Selection method), 211

clear_execution_histories() (raf-
 con.core.state_machine.StateMachine method), 162

clip_buffer() (rafcon.gui.views.logging_console.LoggingConsoleView
 method), 228

Clipboard (class in rafcon.gui.clipboard), 267

close_all_pages() (raf-
 con.gui.controllers.states_editor.StatesEditorController
 method), 192

close_page() (rafcon.gui.controllers.states_editor.StatesEditorController
 method), 192

close_pages_for_specific_sm_id() (raf-
 con.gui.controllers.states_editor.StatesEditorController
 method), 192

code_changed() (raf-
 con.gui.controllers.utils.editor.EditorController
 method), 176

code_changed() (raf-
 con.gui.views.utils.editor.EditorView method), 227

compare_models() (rafcon.gui.action.Action method), 262

compile_module() (rafcon.core.script.Script method), 161

compiled_module (rafcon.core.script.Script property), 161

ComplexAction0bserver (class in raf-
 con.gui.models.state_machine), 216

concurrency_queue (rafcon.core.states.state.State
 property), 146

concurrency_state

module, 131

ConcurrencyItem (class in raf-
 con.core.execution.execution_history_items), 115

ConcurrencyState (class in raf-
 con.core.states.concurrency_state), 131

config

Config (class in rafcon.core.config), 156

connect_button_to_function() (raf-
 con.gui.controllers.menu_bar.MenuBarController
 method), 186

connect_connection_to_port() (raf-
 con.gui.mygaphas.items.state.StateView
 method), 249

connect_signal() (raf-
 con.gui.controllers.utils.extended_controller.ExtendedController
 method), 176

connect_to_income() (raf-
 con.gui.mygaphas.items.state.StateView
 method), 249

connect_to_input_port() (raf-
 con.gui.mygaphas.items.state.StateView
 method), 249

connect_to_outcome() (raf-
 con.gui.mygaphas.items.state.StateView
 method), 249

connect_to_output_port() (raf-
 con.gui.mygaphas.items.state.StateView
 method), 249

connect_to_scoped_variable_port() (raf-
 con.gui.mygaphas.items.state.StateView
 method), 249

connected (rafcon.gui.mygaphas.items.ports.PortView
 property), 247

connected_connections (raf-
 con.gui.mygaphas.items.ports.PortView
 property), 247

connected_incoming (raf-
 con.gui.mygaphas.items.ports.PortView
 property), 247

connected_outgoing (raf-
 con.gui.mygaphas.items.ports.PortView
 property), 247

ConnectionPlaceholderView (class in raf-
 con.gui.mygaphas.items.connection), 245

ConnectionView (class in raf-
 con.gui.mygaphas.items.connection), 245

constants

module, 271

consume() (rafcon.core.execution.consumers.abstract_execution_history
 method), 119

consume() (rafcon.core.execution.consumers.file_system_consumer.FileSys
 method), 119

consumers_exist	(raf- con.core.execution.consumer_manager.ExecutionHistoryContainerManager property), 114	con.gui.controllers.state_editor.scoped_variable_list.ScopedVariableListContainer CORE_ELEMENT_CLASS	(raf- con.gui.controllers.state_editor.transitions.StateTransitionsListContainer attribute), 174
container_state	module, 132	CORE_ELEMENT_CLASS	(raf- con.gui.controllers.state_machine_tree.StateMachineTreeController attribute), 190
ContainerState	(class in con.core.states.container_state), 132	core_element_id	(raf- con.core.state_elements.state_element.StateElement property), 127
ContainerStateModel	(class in con.gui.models.container_state), 204	core_element_id (rafcon.core.states.state.State property), 146	(raf- con.gui.controllers.state_editor.outcomes.StateOutcomesListContainer attribute), 171
convert_if_human_readable()	(raf- con.gui.controllers.library_tree.LibraryTreeController static method), 184	CORE_PARENT_STORAGE_ID	(raf- con.gui.controllers.state_editor.outcomes.StateOutcomesListContainer attribute), 171
convert_string_to_type() (in module raf- con.utils.type_helpers), 279	CORE_STORAGE_ID	(raf- con.gui.controllers.state_editor.outcomes.StateOutcomesListContainer attribute), 171	
convert_string_value_to_type_value() (in mod- ule rafcon.utils.type_helpers), 279	CoreObjectIdentifier (class in rafcon.gui.action), 263		
copy() (rafcon.gui.clipboard.Clipboard method), 267	corner_handles	(raf- con.gui.mygaphas.items.state.StateView property), 249	
copy_file_or_folder() (in module raf- con.utils.filesystem), 272	correct_reference_state()	(raf- con.gui.action.AddObjectAction method), 263	
copy_meta_data_from_state_m()	(raf- con.gui.models.abstract_state.AbstractStateManager method), 199	correct_reference_state()	(raf- con.gui.action.RemoveObjectAction method), 263
copy_meta_data_from_state_m()	(raf- con.gui.models.container_state.ContainerStateModel method), 204	create()	(in module rafcon.utils.multi_event), 277
copy_meta_data_from_state_m()	(raf- con.gui.models.library_state.LibraryStateManager method), 207	create_button()	(in module raf- con.gui.controllers.states_editor), 194
core_data (rafcon.gui.action.StateImage attribute), 266	create_button_label()	(in module raf- con.gui.helpers.label), 231	
core_element (rafcon.gui.models.abstract_state.AbstractStateManager)	create_check_menu_item()	(in module raf- con.gui.helpers.label), 231	
core_element (rafcon.gui.models.data_flow.DataFlowModel)	create_folder_cmd_line()	(in module raf- con.core.interface), 159	
core_element (rafcon.gui.models.data_port.DataPortModel)	create_holder_func()	(in module raf- con.core.interface), 159	
core_element (rafcon.gui.models.logical_port.IncomeModel)	create_label_widget_with_icon()	(in module raf- con.gui.helpers.label), 231	
core_element (rafcon.gui.models.logical_port.OutcomeModel)	create_left_bar_window_title()	(in module raf- con.gui.helpers.label), 231	
core_element (rafcon.gui.models.scoped_variable.ScopedVariableHolder)	create_logger_warning_if_shortcuts_are_overwritten_by_menu- property), 209	(rafcon.gui.controllers.menu_bar.MenuBarController static method), 186	
CORE_ELEMENT_CLASS	create_menu_item()	(in module raf- con.gui.controllers.state_editor.outcomes.StateOutcomesListContainer attribute), 232	
CORE_ELEMENT_CLASS	create_mo_files()	(in module rafcon.utils.i18n), 275	
CORE_ELEMENT_CLASS	create_new_connection()	(in module raf-	

con.gui.mygaphas.utils.gap_helper), 254
create_new_state_from_state_with_type() (in
module rafcon.gui.helpers.state), 234
create_output_dictionary_for_state() (*raf-*
con.core.states.state.State static method), 146
create_path() (in module *rafcon.utils.filesystem*), 272
create_state_from_image() (in module *raf-*
con.gui.action), 267
create_state_image() (in module *rafcon.gui.action*),
267
create_state_model_for_state() (in module *raf-*
con.gui.helpers.state), 234
create_sticky_button() (in module *raf-*
con.gui.controllers.states_editor), 194
create_tab_close_button() (in module *raf-*
con.gui.controllers.states_editor), 194
create_tab_header() (in module *raf-*
con.gui.controllers.states_editor), 194
create_tab_header_label() (in module *raf-*
con.gui.helpers.label), 232
create_text_buffer() (*raf-*
con.gui.views.logging_console.LoggingConsoleVi-
static method), 228
create_widget_title() (in module *raf-*
con.gui.helpers.label), 232
current_history_element (*raf-*
con.gui.models.modification_history.ModificationsHistory *property), 219*
current_state_machine_m (*raf-*
con.gui.controllers.states_editor.StatesEditorController *property), 192*
custom_exceptions
module, 157
CustomCondition (class in *raf-*
con.core.execution.execution_status), 117
cut() (*rafcon.gui.clipboard.Clipboard* method), 268

D

data_flow
module, 120
data_flow (*rafcon.gui.models.data_flow.DataFlowModel* property), 210
data_flow_id (*rafcon.core.state_elements.data_flow.DataFlow* property), 120
data_flow_mode_toggled_shortcut() (*raf-*
con.gui.controllers.menu_bar.MenuBarController method), 186
data_flows (*rafcon.core.states.container_state.ContainerState* property), 134
data_flows (*rafcon.gui.models.container_state.ContainerStateManager* property), 204
data_flows (*rafcon.gui.models.selection.Selection* property), 212

data_port
module, 121
data_port (*rafcon.gui.models.data_port.DataPortModel* property), 210
data_port_id (*rafcon.core.state_elements.data_port.DataPort* property), 122
data_port_type (*raf-*
con.core.state_elements.scope.ScopedData property), 124
data_type (*rafcon.core.state_elements.data_port.DataPort* property), 122
DATA_TYPE_AS_STRING_STORAGE_ID (*raf-*
con.gui.controllers.global_variable_manager.GlobalVariableManager attribute), 181
DATA_TYPE_NAME_STORAGE_ID (*raf-*
con.gui.controllers.state_editor.scoped_variable_list.ScopedVariableList attribute), 166
DataFlow (class in *rafcon.core.state_elements.data_flow*), 120
DataFlowAction (class in *rafcon.gui.action*), 263
DataFlowModel (class in *rafcon.gui.models.data_flow*),
209
DataFlowPlaceholderView (class in *raf-*
con.gui.mygaphas.items.connection), 245
DataFlowView (class in *raf-*
con.gui.mygaphas.items.connection), 245
DataPort (class in *raf-*
con.core.state_elements.data_port), 121
DataPortAction (class in *rafcon.gui.action*), 264
DataPortModel (class in *rafcon.gui.models.data_port*),
210
DataPortView (class in *raf-*
con.gui.mygaphas.items.ports), 246
DeciderState (class in *raf-*
con.core.states.barrier_concurrency_state), 130
decorators
module, 271
default_value (*rafcon.core.state_elements.data_port.DataPort* property), 122
DEFAULT_VALUE_STORAGE_ID (*raf-*
con.gui.controllers.state_editor.scoped_variable_list.ScopedVariableList attribute), 166
deg2rad() (in module *rafcon.utils.geometry*), 273
delete_core_element_of_model() (in module *raf-*
con.gui.helpers.state_machine), 237
delete_core_elements_of_models() (in module *raf-*
con.gui.helpers.state_machine), 237
delete_selected_elements() (in module *raf-*
con.gui.helpers.state_machine), 238
description (*rafcon.core.states.state.State* property),
146
description() (*rafcon.gui.action.AbstractAction* method), 262

description_editor
 module, 170

DescriptionEditorController (class in raf-
 con.gui.controllers.state_editor.description_editor),
 170

DescriptionEditorView (class in raf-
 con.gui.views.state_editor.description_editor),
 226

destroy() (rafcon.core.execution.base_execution_history.BaseExecutionHistory method),
 112

destroy() (rafcon.core.execution.execution_history_items.ConcurrentHistoryItem.controllers.utils.extended_controller.ExtendedController method), 115

destroy() (rafcon.core.execution.execution_history_items.HistoryItem module rafcon.utils.geometry), 273

 method), 116

destroy() (rafcon.core.execution.in_memory_execution_history.InMemoryExecutionHistoryExtendedGtkView method), 260

destroy() (rafcon.core.states.container_state.ContainerState do_delete_item
 method), 134

destroy() (rafcon.core.states.library_state.LibraryState
 method), 142

destroy() (rafcon.core.states.state.State method), 146

destroy() (rafcon.gui.clipboard.Clipboard method),
 268

destroy() (rafcon.gui.controllers.execution_history.ExecutionHistory do_getClipboard
 method), 179

destroy() (rafcon.gui.controllers.logging_console.LoggingConsole do_getClipboard
 method), 183

destroy() (rafcon.gui.controllers.menu_bar.MenuBarController do_getClipboard
 method), 186

destroy() (rafcon.gui.controllers.state_editor.data_flows.DataFlows do_getClipboard
 method), 168

destroy() (rafcon.gui.controllers.state_editor.outcomes.StateOutcomes do_getClipboard
 method), 171

destroy() (rafcon.gui.controllers.state_editor.transitions.TransitionList do_getClipboard
 method), 174

destroy() (rafcon.gui.controllers.utils.extended_controller.Edge do_getClipboard
 method), 176

destroy() (rafcon.gui.models.auto_backup.AutoBackupModel draw()
 method), 222

destroy() (rafcon.gui.models.state_machine.StateMachineModel draw()
 method), 217

destroy() (rafcon.gui.shortcut_manager.ShortcutManager draw()
 method), 269

destroy_all_models_in_dict() (raf-
 con.gui.clipboard.Clipboard static method),
 268

destroy_execution_histories() (raf-
 con.core.state_machine.StateMachine method),
 162

destruction_signal (raf-
 con.gui.models.abstract_state.AbstractStateModel property), 199

destruction_signal (raf-
 con.gui.models.state_element.StateElementModel draw_label_path()
 (in module raf-

 property), 208

destruction_signal (raf-
 con.gui.models.state_machine.StateMachineModel property), 217

diff_related_elements() (raf-
 con.gui.action.RemoveObjectAction method),
 264

Direction (in module raf-
 module raf-

disconnect_all_signals() (raf-
 method), 177

do_configure_event() (raf-
 method), 273

do_selection_reduction_to_one_parent() (raf-
 con.gui.clipboard.Clipboard static method),
 268

do_smart_selection_adaption() (raf-
 method), 268

do_selection_reduction_to_one_parent() (raf-
 method), 246

draw() (rafcon.gui.mygaphas.items.ports.PortView
 method), 247

draw() (rafcon.gui.mygaphas.items.ports.ScopedVariablePortView
 method), 248

draw() (rafcon.gui.mygaphas.items.state.NameView
 method), 248

draw() (rafcon.gui.mygaphas.items.state.StateView
 method), 249

draw_head() (rafcon.gui.mygaphas.items.line.PerpLine
 method), 244

draw_label_path() (in module raf-

con.gui.mygaphas.utils.gap_draw_helper), 252
draw_name() (*rafcon.gui.mygaphas.items.ports.PortView method*), 247
draw_name() (*rafcon.gui.mygaphas.items.ports.ScopedVariablePortView.core.execution.execution_status.ExecutionStatus method*), 248
draw_port() (*rafcon.gui.mygaphas.items.ports.PortView method*), 247
draw_port_label() (*in module rafcon.gui.mygaphas.utils.gap_draw_helper*), 252
draw_tail() (*rafcon.gui.mygaphas.items.line.PerpLine method*), 244
duration (*rafcon.utils.timer.Timer property*), 279

E

editor
module, 175
EditorController (*class in rafcon.gui.controllers.utils.editor*), 175
EditorView (*class in rafcon.gui.views.utils.editor*), 227
ellipsize_labels_recursively() (*in module rafcon.gui.helpers.label*), 232
emit() (*rafcon.utils.log_helpers.LoggingViewHandler method*), 276
emit_undo_redo_signal() (*rafcon.gui.action.Action method*), 262
end_handles() (*rafcon.gui.mygaphas.items.line.PerpLine method*), 244
enforce_generation_of_state_copy_model() (*rafcon.gui.models.library_state.LibraryStateModel method*), 207
enqueue() (*rafcon.core.execution.consumers.abstract_execution.abstract_execution.Consumer method*), 119
enums
module, 158
environment variable
RAFCON_LIBRARY_PATH, 39, 59, 78
RAFCON_LOGGING_CONF, 30, 77, 78
RAFCON_PLUGIN_PATH, 78, 87, 89
RAFCON_START_MINIMIZED, 24, 78
equal() (*in module rafcon.utils.geometry*), 273
exchange_model() (*rafcon.gui.mygaphas.canvas.MyCanvas method*), 256
execute() (*rafcon.core.script.Script method*), 161
execution_engine
module, 115
execution_histories
(rafcon.core.state_machine.StateMachine property), 162
execution_history
module, 117
con.gui.execution_history.Focus (rafcon.gui.controllers.execution_history.ExecutionHistoryTreeController method), 179
execution_mode (*rafcon.core.execution.execution_status.ExecutionStatus property*), 117
execution_state
module, 140
ExecutionHistoryConsumerManager (*class in rafcon.core.execution.consumer_manager*), 114
ExecutionHistoryFactory (*class in rafcon.core.execution.execution_history_factory*), 115
ExecutionHistoryTreeController (*class in rafcon.gui.controllers.execution_history*), 178
ExecutionHistoryTreeView (*class in rafcon.gui.views.execution_history*), 228
ExecutionHistoryView (*class in rafcon.gui.views.execution_history*), 228
ExecutionState (*class in rafcon.core.states.execution_state*), 140
ExecutionStatus (*class in rafcon.core.execution.execution_status*), 117
expected_future_models (*rafcon.gui.models.state.StateModel attribute*), 203
extend_extents() (*in module rafcon.gui.mygaphas.utils.gap_helper*), 254
extend_selection() (*in module rafcon.gui.models.selection*), 214
extended_controller
module, 176
ExtendedController (*class in rafcon.core.execution.history.Consumer con.gui.controllers.utils.extended_controller*), 176
ExtendedGtkView (*class in rafcon.gui.mygaphas.view*), 260
extract_child_models_of_state() (*in module rafcon.gui.helpers.state*), 235
extract_library_properties_from_selected_row() (*rafcon.gui.controllers.library_tree.LibraryTreeController method*), 184

F

feed_consumers() (*rafcon.core.execution.base_execution_history.BaseExecutionHistory method*), 112
FILE_NAME_META_DATA (*in module rafcon.core.storage.storage*), 153
file_system_consumer_exists (*rafcon.core.execution.consumer_manager.ExecutionHistoryConsumer property*), 114
FILE_SYSTEM_CONSUMER_NAME (*rafcon.core.execution.consumer_manager.ExecutionHistoryConsumer property*), 114

attribute), 114
file_system_path (rafcon.core.state_machine.StateMachine property), 162
file_system_path (rafcon.core.states.state.State property), 146
file_system_path (rafcon.gui.action.StateImage attribute), 266
filename (rafcon.core.script.Script property), 161
filesystem module, 272
FileSystemConsumer (class in rafcon.core.execution.consumers.file_system_consumer), 119
filter() (rafcon.utils.log_helpers.NoHigherLevelFilter method), 277
filtered_buffer (rafcon.gui.views.logging_console.LoggingConsoleView property), 228
final_outcome (rafcon.core.states.state.State property), 146
finalize() (rafcon.core.states.state.State method), 146
finalize_backward_execution() (rafcon.core.states.concurrency_state.ConcurrencyState method), 131
finalize_concurrency_state() (rafcon.core.states.concurrency_state.ConcurrencyState method), 131
find_dirty_lock_file_for_state_machine_path() (in module rafcon.gui.models.auto_backup), 222
find_free_and_valid_data_flows() (rafcon.gui.controllers.state_editor.data_flows.StateDataFlowsList class method), 169
find_free_keys() (in module rafcon.gui.controllers.state_editor.data_flows), 169
find_horizontal_guides() (rafcon.gui.mygaphas.guide.GuidedStateMixin method), 260
find_libraries_dependencies() (in module rafcon.gui.helpers.state_machine), 238
find_library_dependencies() (in module rafcon.gui.helpers.state_machine), 238
find_library_dependencies_via_grep() (in module rafcon.core.storage.storage), 154
find_library_root_dependencies() (in module rafcon.gui.helpers.state_machine), 238
find_vertical_guides() (rafcon.gui.mygaphas.guide.GuidedStateMixin method), 260
finish_new_action() (rafcon.gui.models.modification_history.ModificationsHistoryModelProperty), 220
focus (rafcon.gui.models.selection.Selection property), 212
focus_signal (rafcon.gui.models.selection.Selection property), 212
focused_item (rafcon.gui.mygaphas.view.ExtendedGtkView property), 260
format_default_folder_name() (in module rafcon.gui.helpers.text_formatting), 243
format_error_string() (rafcon.gui.controllers.state_editor.source_editor.SourceEditorController static method), 167
format_folder_name_human_readable() (in module rafcon.gui.helpers.text_formatting), 243
free_to_port_external (rafcon.gui.controllers.state_editor.data_flows.StateDataFlowsListController attribute), 169
free_to_port_internal (rafcon.gui.controllers.state_editor.data_flows.StateDataFlowsListController attribute), 169
from_dict() (rafcon.core.state_elements.data_flow.DataFlow class method), 120
from_dict() (rafcon.core.state_elements.data_port.DataPort class method), 122
from_dict() (rafcon.core.state_elements.logical_port.Income class method), 123
from_dict() (rafcon.core.state_elements.logical_port.Outcome class method), 124
from_dict() (rafcon.core.state_elements.scope.ScopedData class method), 125
from_dict() (rafcon.core.state_elements.scope.ScopedVariable class method), 126
from_dict() (rafcon.core.state_elements.state_element.StateElement class method), 127
from_dict() (rafcon.core.state_elements.transition.Transition class method), 126
from_dict() (rafcon.core.state_machine.StateMachine class method), 162
from_dict() (rafcon.core.states.barrier_concurrency_state.BarrierConcurrencyState class method), 129
from_dict() (rafcon.core.states.container_state.ContainerState class method), 134
from_dict() (rafcon.core.states.execution_state.ExecutionState class method), 140
from_dict() (rafcon.core.states.library_state.LibraryState class method), 142
from_dict() (rafcon.core.states.state.State class method), 147
from_dict() (rafcon.utils.vividict.Vividict class method), 280
from_handle() (rafcon.gui.mygaphas.items.line.PerpLine method), 244
from_key (rafcon.core.state_elements.data_flow.DataFlow property), 121
from_outcome (rafcon.core.state_elements.transition.Transition

property), 127

FROM_OUTCOME_STORAGE_ID *(raf-con.gui.controllers.state_editor.transitions.StateTransitionsListClipboard.Clipboard method), 268*

attribute), 174

from_port *(rafcon.gui.mygaphas.items.line.PerpLine property), 244*

from_port_external *(raf-con.gui.controllers.state_editor.data_flows.StateDataFlowsListController.get_all() (rafcon.gui.models.selection.Selection attribute), 169*

from_port_internal *(raf-con.gui.controllers.state_editor.data_flows.StateDataFlowsListController.get_all_ports() (raf-con.gui.mygaphas.items.state.StateView attribute), 169*

from_state *(rafcon.core.state_elements.data_flow.DataFlow property), 121*

from_state *(rafcon.core.state_elements.scope.ScopedData.get_buffer() (rafcon.gui.views.utils.editor.EditorView property), 125*

from_state *(rafcon.core.state_elements.transition.Transition.get_cause_and_affected_model_list() (raf-con.gui.models.container_state.ContainerStateModel property), 127*

FROM_STATE_STORAGE_ID *(raf-con.gui.controllers.state_editor.data_flows.StateDataFlowsListController.get_cause_and_affected_model_list() (raf-con.gui.models.state.StateModel attribute), 168*

FROM_STATE_STORAGE_ID *(raf-con.gui.controllers.state_editor.transitions.StateTransitionListControllers.get_col_rgba() (in module raf-con.gui.mygaphas.utils.gap_draw_helper), 252*

generate_data_flow_id() *(in module raf-con.core.id_generator), 158*

generate_data_port_id() *(in module raf-con.core.id_generator), 158*

generate_linux_launch_files() *(in module raf-con.gui.helpers.state_machine), 238*

generate_outcome_id() *(in module raf-con.core.id_generator), 158*

generate_rafcon_instance_lock_file() *(in module rafcon.gui.models.auto_backup), 222*

generate_right_click_menu() *(raf-con.gui.controllers.library_tree.LibraryTreeController.get_color_active() (raf-con.gui.controllers.modification_history.ModificationHistoryTree static method), 189*

generate_run_id() *(rafcon.core.states.state.State method), 147*

generate_script_id() *(in module raf-con.core.id_generator), 158*

generate_semantic_data_key() *(in module raf-con.core.id_generator), 158*

generate_state_machine_id() *(in module raf-con.core.id_generator), 158*

generate_state_name_id() *(in module raf-con.core.id_generator), 158*

generate_time_stamp() *(in module raf-con.core.state_elements.scope), 126*

generate_transition_id() *(in module raf-con.core.id_generator), 158*

geometry *(raf-con.gui.mygaphas.constraint.PortRectConstraint module), 273*

get_action_arguments() *(raf-con.gui.controllers.state_editor.transitions.StateTransitionsListClipboard.Clipboard method), 268*

get_adjusted_border_positions() *(raf-con.gui.mygaphas.constraint.PortRectConstraint method), 259*

get_all() *(rafcon.gui.models.selection.Selection get_all_ports() (raf-con.gui.mygaphas.items.state.StateView attribute), 169*

get_allowed_state_classes() *(raf-con.gui.controllers.state_editor.overview.StateOverviewController static method), 173*

get_buffer() *(rafcon.gui.views.utils.editor.EditorView method), 227*

get_cause_and_affected_model_list() *(raf-con.gui.models.container_state.ContainerStateModel method), 204*

get_cause_and_affected_model_list() *(raf-con.gui.models.state.StateModel method), 203*

get_connections_for_state() *(raf-con.core.states.container_state.ContainerState method), 134*

get_connections_for_state_and_scoped_variables() *(rafcon.core.states.container_state.ContainerState method), 135*

get_controller() *(raf-con.gui.controllers.utils.extended_controller.ExtendedController method), 177*

get_controller_by_path() *(raf-con.gui.controllers.utils.extended_controller.ExtendedController method), 177*

get_core_data_path() *(in module raf-con.core.storage.storage), 154*

get_current_branch_history_ids() *(raf-con.gui.models.modification_history.ModificationsHistory method), 219*

get_current_state_m() *(raf-con.gui.controllers.states_editor.StatesEditorController method), 192*

get_current_time_string() (in module `raf-con.utils.storage_utils`), 279
get_cursor_position() (raf-
con.gui.views.logging_console.LoggingConsoleView
method), 228
get_cursor_position() (raf-
con.gui.views.utils.editor.EditorView
method), 227
get_data_file_path() (in module `raf-con.utils.resources`), 278
get_data_flow_m() (raf-
con.gui.models.container_state.ContainerStateModel
method), 204
get_data_port() (raf-
con.core.states.container_state.ContainerState
method), 135
get_data_port_by_id() (raf-
con.core.states.container_state.ContainerState
method), 135
get_data_port_by_id() (rafcon.core.states.state.State
method), 147
get_data_port_ids() (raf-
con.core.states.container_state.ContainerState
method), 135
get_data_port_ids() (rafcon.core.states.state.State
method), 147
get_data_port_m() (raf-
con.gui.models.abstract_state.AbstractStateModel
method), 199
get_data_port_m() (raf-
con.gui.models.container_state.ContainerStateModel
method), 204
get_default_config_path() (in module `raf-con.utils.filesystem`), 272
get_default_input_values_for_state() (raf-
con.core.states.state.State method), 147
get_default_log_path() (in module `raf-con.utils.filesystem`), 272
get_element_for_history_id() (raf-
con.gui.models.modification_history.ModificationsHistory
method), 219
get_errors_for_state_name() (raf-
con.core.states.barrier_concurrency_state.Decider
method), 130
get_excluded_items() (raf-
con.gui.mygaphas.guide.GuidedStateMixin
method), 260
get_executed_history_ids() (raf-
con.gui.models.modification_history.ModificationsHistory
method), 219
get_execution_history() (raf-
con.core.execution.execution_history_factory.ExecutionHistoryFactory
static method), 115
get_file_name() (raf-
con.gui.controllers.state_editor.SourceEditorController
method), 167
get_file_system_consumer_file_name() (raf-
con.core.execution.consumer_manager.ExecutionHistoryConsumer
method), 114
get_first_view() (raf-
con.gui.mygaphas.canvas.MyCanvas
method), 256
get_history_item_for_tree_iter() (raf-
con.gui.controllers.execution_history.ExecutionHistoryTreeController
method), 179
get_history_path_from_current_to_target_history_id() (rafcon.gui.models.modification_history.ModificationsHistory
method), 220
get_input_data_port_m() (raf-
con.gui.models.abstract_state.AbstractStateModel
method), 199
get_inputs_for_state() (raf-
con.core.states.container_state.ContainerState
method), 135
get_io_data_port_id_from_name_and_type() (raf-
con.core.states.state.State method), 147
get_items_at_point() (raf-
con.gui.mygaphas.view.ExtendedGtkView
method), 261
get_key_combos() (in module `raf-con.gui.controllers.state_editor.data_flows`), 169
get_label_of_menu_item_box() (in module `raf-con.gui.helpers.label`), 232
get_last_execution_log_filename() (raf-
con.core.state_machine.StateMachine method), 162
get_last_history_item() (raf-
con.core.execution.base_execution_history.BaseExecutionHistory
method), 112
get_last_history_item() (raf-
con.core.execution.in_memory_execution_history.InMemoryExecutionHistory
method), 117
get_line_number_next_to_cursor_with_string_within() (rafcon.gui.views.logging_console.LoggingConsoleView
method), 228
get_logger() (in module `rafcon.utils.log`), 276
get_logic_ports() (raf-
con.gui.mygaphas.items.state.StateView
method), 250
get_menu_item_text() (raf-
con.gui.controllers.library_tree.LibraryTreeController
method), 184
get_meta_data_path() (in module `raf-con.core.storage.storage`), 154
get_modify_info() (rafcon.gui.models.state.StateModel
method), 203
get_modification_lock() (raf-

```

    con.core.state_machine.StateMachine method),           method), 210
    163
get_new_list_store()                               (raf- get_observable_destruction_signal()      (raf-
    con.gui.controllers.state_editor.scoped_variable_list.ScopedVariableListController
    static method), 166
get_next_element()                                (raf- get_observable_destruction_signal()      (raf-
    con.gui.models.modification_history.ModificationsHistory method), 208
    method), 220
get_next_upper_library_root_state()               (raf- get_observable_destruction_signal()      (raf-
    con.core.states.state.State method), 147
get_notebook_tab_title()  (in module raf- get_observable_focus_signal()      (raf-
    con.gui.helpers.label), 232
get_number_of_data_flows()                         (raf- con.gui.models.selection.Selection method),
    con.core.states.container_state.ContainerState
    method), 136
get_number_of_data_flows()                         (raf- get_observable_income()          (raf-
    con.core.states.library_state.LibraryState
    method), 142
get_number_of_data_flows()                         (raf- get_observable_income()          (raf-
    con.core.states.state.State method), 147
get_number_of_transitions()                       (raf- con.gui.models.logical_port.IncomeModel
    con.core.states.container_state.ContainerState
    method), 136
get_number_of_transitions()                       (raf- method), 215
get_number_of_transitions()                       (raf- get_observable_input_data_ports() (raf-
    con.core.states.library_state.LibraryState
    method), 142
get_number_of_transitions()                       (raf- con.gui.models.abstract_state.AbstractStateModel
    method), 199
get_number_of_transitions()                       (raf- get_observable_is_start()        (raf-
    con.core.states.state.State method), 148
get_number_of_waiting_threads()                  (raf- con.gui.models.abstract_state.AbstractStateModel
    con.core.execution.execution_status.CustomCondition
    method), 208
get_object_hash_string()                          (raf- get_observable_meta_signal()      (raf-
    con.utils.hashable.Hashable static method),
    274
get_object_identifier()                           (raf- con.gui.models.state_machine.StateMachineModel
    con.gui.action.RemoveObjectAction method),
    265
get_observable_action_signal()                   (raf- method), 217
    con.gui.models.abstract_state.AbstractStateModel
    method), 199
get_observable_action_signal()                   (raf- get_observable_ongoing_complex_actions() (raf-
    con.gui.models.state_machine.StateMachineModel
    method), 217
get_observable_change_count()                   (raf- get_observable_outcome()         (raf-
    con.gui.models.modification_history.ModificationsHistory
    method), 220
    made by con.gui.models.abstract_state.AbstractStateModel
    method), 199
get_observable_data_flow()                      (raf- get_observable_outcomes()        (raf-
    con.gui.models.data_flow.DataFlowModel
    method), 210
get_observable_data_flows()                     (raf- get_observable_output_data_ports() (raf-
    con.gui.models.container_state.ContainerStateModel
    method), 205
get_observable_data_port()                      (raf- con.gui.models.abstract_state.AbstractStateModel
    con.gui.models.data_port.DataPortModel
    method), 199
get_observable_root_state()                    (raf- get_observable_scoped_variable() (raf-
    con.gui.models.state_machine.StateMachineModel
    method), 217
get_observable_scoped_variable()              (raf- con.gui.models.scoped_variable.ScopedVariableModel
    con.gui.models.scoped_variable.ScopedVariableModel
    method), 189

```

<code>method), 211</code>	<code>get_port_area()</code>	<code>(raf-</code>
<code>get_observable_scoped_variables()</code> (raf- <code>con.gui.models.container_state.ContainerStateModel</code> <code>method), 205</code>	<code>con.gui.mygaphas.items.ports.PortView</code> <code>method), 247</code>	<code>method), 247</code>
<code>get_observable_selection_changed_signal()</code> (rafcon.gui.models.selection.Selection <code>method), 212</code>	<code>get_port_at_point()</code> (raf- <code>con.gui.mygaphas.view.ExtendedGtkView</code> <code>method), 261</code>	<code>method), 261</code>
<code>get_observable_sm_selection_changed_signal()</code> (rafcon.gui.models.state_machine.StateMachine <code>method), 217</code>	<code>get_port_for_handle()</code> (in module raf- <code>con.gui.mygaphas.utils.gap_helper), 254</code>	<code>con.gui.mygaphas.items.state.StateView</code> <code>method), 250</code>
<code>get_observable_state()</code> (raf- <code>con.gui.models.abstract_state.AbstractStateModel</code> <code>method), 199</code>	<code>get_possible_combos_for_transition()</code> (raf- <code>con.gui.controllers.state_editor.transitions.StateTransitionsListCo</code> <code>static method), 174</code>	<code>method), 199</code>
<code>get_observable_state_action_signal()</code> (raf- <code>con.gui.models.state_machine.StateMachineMode</code> <code>method), 217</code>	<code>get_previous_element()</code> (raf- <code>con.gui.models.modification_history.ModificationsHistory</code> <code>method), 220</code>	<code>method), 217</code>
<code>get_observable_state_machine()</code> (raf- <code>con.gui.models.state_machine.StateMachineMode</code> <code>method), 217</code>	<code>get_previously_executed_state()</code> (raf- <code>con.core.states.state.State method), 148</code>	<code>method), 217</code>
<code>get_observable_state_meta_signal()</code> (raf- <code>con.gui.models.state_machine.StateMachineModel</code> <code>method), 217</code>	<code>get_relative_positions_of_waypoints()</code> (in mod- ule rafcon.gui.mygaphas.utils.gap_helper), 254	<code>method), 217</code>
<code>get_observable_states()</code> (raf- <code>con.gui.models.container_state.ContainerStateModel</code> <code>method), 205</code>	<code>get_repository_share_path()</code> (in module raf- <code>con.utils.resources), 278</code>	<code>method), 205</code>
<code>get_observable_transition()</code> (raf- <code>con.gui.models.transition.TransitionModel</code> <code>method), 209</code>	<code>get_root_state_description_of_sm_file_system_path()</code> (in module rafcon.gui.helpers.state_machine), <code>239</code>	<code>method), 209</code>
<code>get_observable_transitions()</code> (raf- <code>con.gui.models.container_state.ContainerStateModel</code> <code>method), 205</code>	<code>get_root_state_element_meta()</code> (raf- <code>con.gui.models.modification_history.ModificationsHistoryModel</code> <code>method), 220</code>	<code>method), 205</code>
<code>get_outcome()</code> (rafcon.core.states.container_state.ContainerState <code>method), 136</code>	<code>get_root_state_file_path()</code> (in module raf- <code>con.gui.helpers.state_machine), 239</code>	<code>method), 136</code>
<code>get_outcome_for_state_id()</code> (raf- <code>con.core.states.barrier_concurrency_state.Decide</code> <code>method), 130</code>	<code>get_root_state_name_of_sm_file_system_path()</code> (in module rafcon.gui.helpers.state_machine), <code>239</code>	<code>method), 130</code>
<code>get_outcome_for_state_name()</code> (raf- <code>con.core.states.barrier_concurrency_state.Decide</code> <code>method), 131</code>	<code>get_root_window()</code> (raf- <code>con.gui.controllers.utils.extended_controller.ExtendedController</code> <code>method), 177</code>	<code>method), 131</code>
<code>get_outcome_m()</code> (raf- <code>con.gui.models.abstract_state.AbstractStateModel</code> <code>method), 199</code>	<code>get_root_iter_for_state_model()</code> (raf- <code>con.gui.controllers.state_machine_tree.StateMachineTreeControl</code> <code>method), 190</code>	<code>method), 199</code>
<code>get_output_data_port_m()</code> (raf- <code>con.gui.models.abstract_state.AbstractStateModel</code> <code>method), 200</code>	<code>get_scoped_variable_from_name()</code> (raf- <code>con.core.states.container_state.ContainerState</code> <code>method), 136</code>	<code>method), 200</code>
<code>get_page_of_state_m()</code> (raf- <code>con.gui.controllers.states_editor.StatesEditorCont</code> <code>method), 192</code>	<code>get_scoped_variable_m()</code> (raf- <code>con.gui.models.container_state.ContainerStateModel</code> <code>method), 205</code>	<code>method), 192</code>
<code>get_parent()</code> (rafcon.gui.mygaphas.canvas.MyCanvas <code>method), 256</code>	<code>get_selected_elements_of_core_class()</code> (raf- <code>con.gui.models.selection.Selection</code> <code>method), 212</code>	<code>method), 256</code>
<code>get_parent_state_v()</code> (raf- <code>con.gui.mygaphas.items.line.PerpLine</code> <code>method), 244</code>	<code>get_selected_state()</code> (raf- <code>con.gui.models.selection.Selection</code> <code>method), 212</code>	<code>method), 244</code>
<code>get_path()</code> (rafcon.core.states.state.State <code>method), 148</code>	<code>get_semantic_data()</code> (rafcon.core.states.state.State <code>method), 148</code>	<code>method), 148</code>

get_semantic_dictionary_list() (raf-
 con.gui.clipboard.Clipboard method), 268

get_set_of_arguments() (raf-
 con.gui.action.DataFlowAction static method),
 263

get_set_of_arguments() (raf-
 con.gui.action.DataPortAction static method),
 264

get_set_of_arguments() (raf-
 con.gui.action.OutcomeAction static method),
 264

get_set_of_arguments() (raf-
 con.gui.action.StateAction static method),
 265

get_set_of_arguments() (raf-
 con.gui.action.StateElementAction static
 method), 266

get_set_of_arguments() (raf-
 con.gui.action.TransitionAction static method),
 267

get_side_length_of_resize_handle() (in module
 rafcon.gui.mygaphas.utils.gap_draw_helper),
 252

get_start_state() (raf-
 con.core.states.container_state.ContainerState
 method), 136

get_state_at_point() (raf-
 con.gui.mygaphas.view.ExtendedGtkView
 method), 261

get_state_by_path() (raf-
 con.core.state_machine.StateMachine method),
 163

get_state_changed() (rafcon.gui.action.Action
 method), 263

get_state_drawing_area() (raf-
 con.gui.mygaphas.items.state.StateView static
 method), 250

get_state_element_meta() (in module raf-
 con.gui.action), 267

get_state_element_meta_from_internal_tmp_storage() (rafcon.gui.models.modification_history.ModificationsHistoryModel method), 220

get_state_for_transition() (raf-
 con.core.states.container_state.ContainerState
 method), 136

get_state_identifier() (raf-
 con.gui.controllers.states_editor.StatesEditorController
 method), 193

get_state_identifier_for_page() (raf-
 con.gui.controllers.states_editor.StatesEditorController
 method), 193

get_state_image() (rafcon.gui.action.AbstractAction
 method), 262

get_state_image() (rafcon.gui.action.Action method), 263

get_state_image() (raf-
 con.gui.action.MetaDataAction
 method), 264

get_state_image() (raf-
 con.gui.action.StateElementAction
 method), 266

get_state_machine() (rafcon.core.states.state.State
 method), 148

get_state_machine_m() (raf-
 con.gui.models.abstract_state.AbstractStateModel
 method), 200

get_state_machine_m() (raf-
 con.gui.models.state_element.StateElementModel
 method), 208

get_state_machine_selection() (raf-
 con.gui.controllers.state_machine_tree.StateMachineTreeController
 method), 190

get_state_model() (in module raf-
 con.gui.controllers.state_editor.data_flows),
 169

get_state_model_by_path() (raf-
 con.gui.models.state_machine.StateMachineModel
 method), 217

get_state_model_changed() (raf-
 con.gui.action.MetaDataAction
 method), 264

get_state_model_class_for_state() (in module
 rafcon.gui.models.abstract_state), 202

get_states_statistics() (raf-
 con.core.states.container_state.ContainerState
 method), 136

get_states_statistics() (raf-
 con.core.states.library_state.LibraryState
 method), 142

get_states_statistics() (raf-
 con.core.states.state.State method), 148

get_storage_id_for_state() (in module raf-
 con.core.storage.storage), 154

get_storage_path() (raf-
 con.core.states.library_state.LibraryState
 method), 142

get_storage_path() (rafcon.core.states.state.State
 method), 148

get_temp_file_system_path() (raf-
 con.core.states.state.State method), 149

get_text() (rafcon.gui.views.utils.editor.EditorView
 method), 227

get_text_layout() (in module raf-
 con.gui.mygaphas.utils.gap_draw_helper),
 253

get_text_of_line() (raf-
 con.gui.views.logging_console.LoggingConsoleView
 method), 228

get_transition_for_outcome() (raf-
 con.core.states.container_state.ContainerState
 method), 136

get_transition_m() (raf-
 con.gui.models.container_state.ContainerStateModel
 method), 205

get_transitions() (raf-
 con.gui.mygaphas.items.state.StateView
 method), 250

get_uppermost_library_root_state() (raf-
 con.core.states.state.State method), 149

get_view_for_core_element() (raf-
 con.gui.mygaphas.canvas.MyCanvas
 method), 256

get_view_for_model() (raf-
 con.gui.mygaphas.canvas.MyCanvas
 method), 257

get_widget_title() (in module raf-
 con.gui.helpers.label), 232

get_zoom_factor() (raf-
 con.gui.mygaphas.view.ExtendedGtkView
 method), 261

global_variable_id_generator() (in module raf-
 con.core.id_generator), 158

global_variable_is_editable() (raf-
 con.gui.controllers.global_variable_manager.GlobalVariableManager
 method), 182

global_variable_manager
 module, 158

GlobalVariableEditorView (class in raf-
 con.gui.views.global_variable_editor), 228

GlobalVariableManagerController (class in raf-
 con.gui.controllers.global_variable_manager), 181

glue() (rafcon.gui.mygaphas.connector.RectanglePointPort
 method), 257

glue() (rafcon.gui.mygaphas.guide.GuidedStateHandleInMotion
 method), 259

go_to_history_element() (raf-
 con.gui.models.modification_history.ModificationsHistory
 method), 220

graphical_editor (raf-
 con.gui.mygaphas.view.ExtendedGtkView
 property), 261

group_selected_states_and_scoped_variables() (in module rafcon.gui.helpers.state_machine), 239

group_states() (raf-
 con.core.states.container_state.ContainerState
 method), 137

group_states() (raf-
 con.gui.models.container_state.ContainerStateModel
 method), 205

group_states_and_scoped_variables() (in module rafcon.gui.helpers.state), 235

GuidedNameInMotion (class in raf-
 con.gui.mygaphas.guide), 259

GuidedStateHandleInMotion (class in raf-
 con.gui.mygaphas.guide), 259

GuidedStateInMotion (class in raf-
 con.gui.mygaphas.guide), 259

GuidedStateMixin (class in raf-
 con.gui.mygaphas.guide), 259

H

handle_new_selection() (raf-
 con.gui.models.selection.Selection
 method), 212

handle_no_start_state() (raf-
 con.core.states.container_state.ContainerState
 method), 137

handle_no_transition() (raf-
 con.core.states.container_state.ContainerState
 method), 137

handle_pos (rafcon.gui.mygaphas.items.ports.PortView
 property), 247

handle_variable_selection_of_core_class_elements() (rafcon.gui.models.selection.Selection
 method), 212

handles() (rafcon.gui.mygaphas.items.ports.PortView
 method), 247

has_label() (rafcon.gui.mygaphas.items.ports.DataPortView
 method), 246

has_label() (rafcon.gui.mygaphas.items.ports.OutcomeView
 method), 246

has_label() (rafcon.gui.mygaphas.items.ports.PortView
 method), 247

has_outgoing_connection() (raf-
 con.gui.mygaphas.items.ports.PortView
 method), 247

hashable (class in rafcon.utils.hashable), 274

height (rafcon.gui.mygaphas.connector.RectanglePointPort
 property), 257

hide_window() (rafcon.gui.controllers.undocked_window.UndockedWindow
 method), 197

hierarchy_level (raf-
 con.gui.models.abstract_state.AbstractStateModel
 property), 200

hierarchy_state
 module, 140

HierarchyState (class in raf-
 con.core.states.hierarchy_state), 140

history_id (rafcon.gui.models.modification_history.HistoryTreeElement
 property), 219

history_item_id_generator() (in module rafcon.core.id_generator), 159

HISTORY_ITEM_STORAGE_ID (rafcon.gui.controllers.execution_history.ExecutionHistoryTree attribute), 179

HistoryItem (class in rafcon.core.execution.execution_history_items), 115

HistoryTreeElement (class in rafcon.gui.models.modification_history), 219

HistoryTreeView (class in rafcon.gui.views.modification_history), 230

hovered (rafcon.gui.mygaphas.items.state.StateView property), 250

hovered_handle (rafcon.gui.mygaphas.view.ExtendedGtkView attribute), 261

|

icons (rafcon.gui.views.state_editor.state_editor.StateEditorView attribute), 224

id() (rafcon.core.states.state.State method), 149

id_generator module, 158

ID_STORAGE_ID (rafcon.gui.controllers.global_variable_manager.GlobalVariableManager attribute), 181

ID_STORAGE_ID (rafcon.gui.controllers.library_tree.LibraryTreeController attribute), 183

ID_STORAGE_ID (rafcon.gui.controllers.state_editor.data_flows.StateDataFlowsListController attribute), 168

ID_STORAGE_ID (rafcon.gui.controllers.state_editor.outcomes.StateOutcomesListController attribute), 171

ID_STORAGE_ID (rafcon.gui.controllers.state_editor.scoped_insertions.ScopedVariableListController attribute), 166

ID_STORAGE_ID (rafcon.gui.controllers.state_editor.transitions.StateTransitionsListController attribute), 174

ID_STORAGE_ID (rafcon.gui.controllers.state_machine_tree.StateMachineTreeController attribute), 190

Income (class in rafcon.core.state_elements.logical_port), 123

income (rafcon.core.states.state.State property), 149

income (rafcon.gui.models.abstract_state.AbstractStateModel property), 200

income (rafcon.gui.models.logical_port.IncomeModel property), 215

income (rafcon.gui.models.selection.Selection property), 213

income (rafcon.gui.mygaphas.items.state.StateView property), 250

IncomeModel (class in rafcon.gui.models.logical_port), 215

incomes (rafcon.gui.models.selection.Selection property), 213

IncomeView (class in rafcon.gui.mygaphas.items.ports), 246

initiate_library_root_state_model() (rafcon.core.models.library_state.LibraryStateModel method), 207

InMemoryExecutionHistory (class in rafcon.core.execution.in_memory_execution_history), 117

input_data (rafcon.core.states.state.State property), 149

input_data_port_runtime_values (rafcon.core.states.library_state.LibraryState property), 142

input_data_ports (rafcon.core.states.state.State property), 149

input_data_ports (rafcon.gui.models.abstract_state.AbstractStateModel property), 200

input_data_ports (rafcon.gui.models.selection.Selection property), 213

input_port() (rafcon.gui.mygaphas.items.state.StateView method), 250

InputDataPort (class in rafcon.core.states.state.State), 122

InputPortsListView (class in rafcon.gui.views.state_editor.input_port_list), 225

InputPortView (class in rafcon.gui.mygaphas.items.state.StateView property), 246

inputs (rafcon.gui.mygaphas.items.state.StateView property), 250

insert_action() (rafcon.core.execution_history.ModificationsHistory method), 320

insert_and_update_recursively() (rafcon.core.execution_history.ExecutionHistoryTreeController method), 190

insert_button_clicked() (rafcon.gui.controllers.library_tree.LibraryTreeController method), 184

insert_concurrent_execution_histories() (rafcon.core.execution_history.ExecutionHistoryTreeController method), 179

insert_execution_history() (rafcon.core.execution_history.ExecutionHistoryTreeController method), 180

insert_history_item() (rafcon.core.execution_history.ExecutionHistoryTreeController method), 180

insert_meta_data_from_models_dict() (rafcon.gui.models.container_state.ContainerStateModel method), 205

insert_rec() (rafcon.gui.controllers.library_tree.LibraryTreeController

method), 184
insert_scoped_variables_tab() (raf-
con.gui.views.state_editor.state_editor.StateEditorView
method), 224
insert_source_tab() (raf-
con.gui.views.state_editor.state_editor.StateEditorView
method), 224
insert_state_as() (in module raf-
con.gui.helpers.state, 235
insert_state_into_selected_state() (in module
rafcon.gui.helpers.state_machine, 239
insert_state_meta_data() (in module raf-
con.gui.action, 267
install_fonts() (in module *rafcon.utils.installation*),
 275
install_locally_required_files() (in module *raf-
 con.utils.installation*), 275
installed_font_faces_for_font() (in module *raf-
 con.utils.installation*), 275
interface
 module, 159
is_about_to_be_destroyed_recursively (raf-
con.gui.models.abstract_state.AbstractStateModel
property), 200
is_about_to_be_destroyed_recursively (raf-
con.gui.models.library_state.LibraryStateModel
property), 207
is_config_loaded_from_file() (raf-
con.core.config.ObservableConfig method),
 157
is_dummy (*rafcon.core.states.container_state.ContainerState*
property), 137
is_event_of_key_string() (in module raf-
con.gui.helpers.label, 232
IS_EXTERNAL_STORAGE_ID (raf-
con.gui.controllers.state_editor.data_flows.StateDataFlowListController
attribute), 168
IS_EXTERNAL_STORAGE_ID (raf-
con.gui.controllers.state_editor.transitions.StateTransitionsListController
attribute), 174
is_in_port() (*rafcon.gui.mygaphas.items.line.PerpLine*
static method), 244
IS_LOCKED_AS_STRING_STORAGE_ID (raf-
con.gui.controllers.global_variable_manager.GlobalVariableManagerController
attribute), 181
is_out_port() (*rafcon.gui.mygaphas.items.line.PerpLine*
static method), 244
is_redo_possible() (raf-
con.gui.models.modification_history.ModificationHistory
method), 220
is_root_state (*rafcon.core.states.state.State* property),
 149
is_root_state_of_library (raf-
con.core.states.state.State property), 149

is_selected() (rafcon.gui.models.selection.Selection
method), 213
is_selected() (rafcon.gui.mygaphas.items.ports.PortView
method), 247
is_selection_inside_of_library_state() (in
module rafcon.gui.helpers.state_machine, 239
is_start (*rafcon.gui.models.abstract_state.AbstractStateModel*
property), 200
is_state_machine_stopped_to_proceed() (in mod-
ule rafcon.gui.helpers.state_machine, 239
is_undo_possible() (raf-
con.gui.models.modification_history.ModificationsHistory
method), 220
item (*rafcon.gui.mygaphas.segment.TransitionSegment*
property), 260
ITEM_STORAGE_ID (raf-
con.gui.controllers.library_tree.LibraryTreeController
attribute), 183
ItemProjection (class in *rafcon.gui.mygaphas.canvas*),
 256

J

join() (*rafcon.core.state_machine.StateMachine*
method), 163
join() (*rafcon.core.states.state.State* method), 149
join_state() (*rafcon.core.states.concurrency_stateConcurrencyState*
method), 131

K

KeepPointWithinConstraint (class in raf-
con.gui.mygaphas.constraint, 258
KeepPortDistanceConstraint (class in raf-
con.gui.mygaphas.constraint, 258
KeepRectangleWithinConstraint (class in raf-
con.gui.mygaphas.constraint, 258
KeepRelativePositionConstraint (class in raf-
con.gui.mygaphas.constraint, 258
keys (*rafcon.core.config.ObservableConfig* property),
attribute), 156
keys_requiring_restart (*rafcon.core.config.Config*
attribute), 156
keys_requiring_restart (raf-
con.core.config.ObservableConfig attribute),
 157
keys_requiring_state_machine_refresh (raf-
con.core.config.ObservableConfig attribute),
 157

LEFT (*rafcon.gui.mygaphas.utils.enums.SnappedSide* at-
tribute), 251
len() (*rafcon.gui.views.logging_console.LoggingConsoleView*
method), 229

LIB_KEY_STORAGE_ID (raf-
con.gui.controllers.library_tree.LibraryTreeController.attribute), 183
LIB_PATH_STORAGE_ID (raf-
con.gui.controllers.library_tree.LibraryTreeController.attribute), 183
library_hierarchy_depth (raf-
con.core.states.library_state.LibraryState.property), 143
library_manager module, 161
library_name (*rafcon.core.states.library_state.LibraryState*.property), 143
library_path (*rafcon.core.states.library_state.LibraryState*.property), 143
library_state module, 141
library_tree module, 183
LibraryNotFoundException, 157
LibraryNotFoundSkipException, 157
LibraryState (class in *rafcon.core.states.library_state*), 141
LibraryStateManager (class in *rafcon.gui.models.library_state*), 207
LibraryTreeController (class in *rafcon.gui.controllers.library_tree*), 183
LibraryTreeView (class in *rafcon.gui.views.library_tree*), 228
limit_pos() (*rafcon.gui.mygaphas.constraint.PortRectConstraint*.static method), 259
limit_string() (in module *rafcon.gui.helpers.text_formatting*), 243
limit_text_max_length() (in module *rafcon.core.storage.storage*), 154
limit_text_to_be_path_element() (in module *rafcon.core.storage.storage*), 154
limit_value_string_length() (in module *rafcon.gui.mygaphas.utils.gap_draw_helper*), 253
linkage_overview module, 165
LinkageOverviewController (class in *rafcon.gui.controllers.state_editor.linkage_overview*), 165
LinkageOverviewDataView (class in *rafcon.gui.views.state_editor.linkage_overview*), 224
LinkageOverviewLogicView (class in *rafcon.gui.views.state_editor.linkage_overview*), 224
LinkageOverviewView (class in *rafcon.gui.views.state_editor.linkage_overview*), 224
load() (*rafcon.core.config.Config* method), 156
load_and_set_file_content() (*rafcon.gui.controllers.state_editor.source_editor.SourceEditorController*.method), 167
load_data_file() (in module *rafcon.core.storage.storage*), 155
load_dict_from_yaml() (in module *rafcon.utils.storage_utils*), 279
load_meta_data() (raf-
con.gui.models.abstract_state.AbstractStateModel.method), 200
load_meta_data() (raf-
con.gui.models.state_machine.StateMachineModel.method), 217
load_objects_from_json() (in module *rafcon.utils.storage_utils*), 279
load_plugins() (in module *rafcon.utils.plugins*), 278
load_state_recursively() (in module *rafcon.core.storage.storage*), 155
log module, 275
log_exceptions (class in *rafcon.utils.log*), 276
log_to_collapsed_structure() (in module *rafcon.utils.execution_log*), 272
log_to_DataFrame() (in module *rafcon.utils.execution_log*), 271
log_to_ganttpplot() (in module *rafcon.utils.execution_log*), 272
log_to_raw_structure() (in module *rafcon.utils.execution_log*), 272
LoggingConsoleController (class in *rafcon.gui.controllers.logging_console*), 183
LoggingConsoleView (class in *rafcon.gui.views.logging_console*), 228
LoggingViewHandler (class in *rafcon.utils.log_helpers*), 276
LogicalPort (class in *rafcon.core.state_elements.logical_port*), 123
LogicalPortModel (class in *rafcon.gui.models.logical_port*), 215
LogicPortView (class in *rafcon.gui.mygaphas.items.ports*), 246
M
MainWindowView (class in *rafcon.gui.views.main_window*), 229
make_file_executable() (in module *rafcon.utils.filesystem*), 272
make_tarfile() (in module *rafcon.utils.filesystem*), 273
MARGIN (*rafcon.gui.mygaphas.guide.GuidedStateMixin*.attribute), 260
marked_dirty (*rafcon.core.state_machine.StateMachine*.property), 163
measure_time() (in module *rafcon.utils.timer*), 279

menu_bar
 module, 185
menu_item_add_library_root_clicked() (raf-
 con.gui.controllers.library_tree.LibraryTreeController
 method), 184
menu_item_find_usages_clicked() (raf-
 con.gui.controllers.library_tree.LibraryTreeController
 method), 184
menu_item_relocate_libraries_or_root_clicked()
 (rafcon.gui.controllers.library_tree.LibraryTreeController
 method), 184
menu_item_remove_libraries_or_root_clicked()
 (rafcon.gui.controllers.library_tree.LibraryTreeController
 method), 184
menu_item_rename_libraries_or_root_clicked()
 (rafcon.gui.controllers.library_tree.LibraryTreeController
 method), 184
MenuBarController (class in raf-
 con.gui.controllers.menu_bar), 185
MenuBarView (class in rafcon.gui.views.menu_bar), 230
meta (rafcon.gui.models.state_machine.StateMachineModel
 attribute), 218
meta_changed()
 (raf-
 con.gui.models.abstract_state.AbstractStateModel
 method), 200
meta_changed()
 (raf-
 con.gui.models.state_element.StateElementModel
 method), 208
meta_changed()
 (raf-
 con.gui.models.state_machine.StateMachineModel
 method), 218
meta_changed_notify_after()
 (raf-
 con.gui.models.modification_history.ModificationsHistory
 method), 221
meta_data (rafcon.gui.action.StateImage attribute), 266
meta_dump_or deepcopy() (in module raf-
 con.gui.action), 267
meta_signal (rafcon.gui.models.abstract_state.AbstractStateModel
 property), 201
meta_signal (rafcon.gui.models.state_element.StateElementModel
 property), 208
meta_signal (rafcon.gui.models.state_machine.StateMachineModel
 property), 218
MetaDataTable (class in rafcon.gui.action), 264
mirror_waypoints() (in module raf-
 con.gui.models.transition), 209
mirror_y_axis_in_vividict_element() (in module
 rafcon.gui.models.abstract_state), 202
missing_library_meta_data (raf-
 con.core.states.container_state.ContainerState
 property), 137
model (rafcon.gui.mygaphas.items.connection.DataFlowView
 property), 245
model (rafcon.gui.mygaphas.items.connection.TransitionView
 model property), 246
model (rafcon.gui.mygaphas.items.ports.DataPortView
 property), 246
model (rafcon.gui.mygaphas.items.ports.IncomeView
 property), 246
model (rafcon.gui.mygaphas.items.ports.OutcomeView
 property), 247
model (rafcon.gui.mygaphas.items.ports.ScopedVariablePortView
 property), 248
model (rafcon.gui.mygaphas.items.state.NameView prop-
 erty), 248
model (rafcon.gui.mygaphas.items.state.StateView prop-
 erty), 250
model_changed()
 (raf-
 con.gui.controllers.library_tree.LibraryTreeController
 method), 184
model_changed()
 (raf-
 con.gui.controllers.logging_console.LoggingConsoleController
 method), 183
model_changed()
 (raf-
 con.gui.models.abstract_state.AbstractStateModel
 method), 201
model_changed()
 (raf-
 con.gui.models.container_state.ContainerStateModel
 method), 205
model_changed()
 (raf-
 con.gui.models.data_flow.DataFlowModel
 method), 210
model_changed()
 (raf-
 con.gui.models.data_port.DataPortModel
 method), 210
model_changed()
 (raf-
 con.gui.models.library_state.LibraryStateModel
 method), 207
model_changed()
 (raf-
 con.gui.models.logical_port.IncomeModel
 method), 215
model_changed()
 (raf-
 con.gui.models.logical_port.OutcomeModel
 method), 216
model_changed()
 (raf-
 con.gui.models.scoped_variable.ScopedVariableModel
 method), 211
model_changed()
 (raf-
 con.gui.models.state.StateModel
 method), 203
model_changed()
 (raf-
 con.gui.models.state_element.StateElementModel
 method), 208
model_changed()
 (raf-
 con.gui.models.transition.TransitionModel
 method), 209
MODEL_STORAGE_ID
 (raf-
 con.gui.controllers.global_variable_manager.GlobalVariableMan-
 attribute), 181

MODEL_STORAGE_ID (raf-
con.gui.controllers.state_editor.data_flows.StateDataFlow, 168
attribute), 168

MODEL_STORAGE_ID (raf-
con.gui.controllers.state_editor.outcomes.StateOutcome, 171
attribute), 171

MODEL_STORAGE_ID (raf-
con.gui.controllers.state_editor.scoped_variable_list.ScopedVariableList, 166
attribute), 166

MODEL_STORAGE_ID (raf-
con.gui.controllers.state_editor.transitions.StateTransition, 174
attribute), 174

MODEL_STORAGE_ID (raf-
con.gui.controllers.state_machine_tree.StateMachineTree, 190
attribute), 190

modification_history
module, 219

modification_lock() (raf-
con.core.state_machine.StateMachine method), 163

ModificationHistoryTreeController (class in raf-
con.gui.controllers.modification_history), 189

ModificationHistoryView (class in raf-
con.gui.views.modification_history), 230

modifications (*rafcon.gui.models.modification_history.ModificationHistory* class, 221
property), 221

ModificationsHistory (class in raf-
con.gui.models.modification_history), 219

ModificationsHistoryModel (class in raf-
con.gui.models.modification_history), 220

modify_origin() (raf-
con.core.state_elements.data_flow.DataFlow method), 121

modify_origin() (raf-
con.core.state_elements.transition.Transition method), 127

modify_target() (raf-
con.core.state_elements.data_flow.DataFlow method), 121

modify_target() (raf-
con.core.state_elements.transition.Transition method), 127

module
barrier_concurrency_state, 129
concurrency_state, 131
config, 156
constants, 271
container_state, 132
custom_exceptions, 157
data_flow, 120
data_port, 121
decorators, 271
description_editor, 170
editor, 175

(raf-
enums, 158
execution_history, 115
execution_state, 140
filesystem, 272
geometry, 273
GlobalVariableManager, 158
hierarchy_state, 140
id_generator, 158
InterfaceController, 176
library_manager, 161
library_state, 141
LinkHistoryTree, 183
linkage_overview, 165
log, 275
menu_bar, 185
modification_history, 219
multi_event, 277
outcome, 123
plugins, 278
preemptive_concurrency_state, 144
rafcon.core.config, 156
rafcon.core.constants, 158
rafcon.core.HistoryManager, 157
rafcon.core.execution.base_execution_history, 112
rafcon.core.execution.consumer_manager, 114
rafcon.core.execution.consumers.abstract_execution_history, 119
rafcon.core.execution.consumers.file_system_consumer, 119
rafcon.core.execution.execution_engine, 115
rafcon.core.execution.execution_history_factory, 115
rafcon.core.execution.execution_history_items, 115
rafcon.core.execution.execution_status, 117
rafcon.core.execution.in_memory_execution_history, 117
rafcon.core.global_variable_manager, 158
rafcon.core.id_generator, 158
rafcon.core.interface, 159
rafcon.core.library_manager, 161
rafcon.core.script, 161
rafcon.core.singleton, 162
rafcon.core.state_elements.data_flow, 120
rafcon.core.state_elements.data_port, 121
rafcon.core.state_elements.logical_port, 123
rafcon.core.state_elements.scope, 124

rafcon.core.state_elements.state_element, 197
rafcon.core.state_elements.transition, 126
rafcon.core.state_machine, 162
rafcon.core.state_machine_manager, 164
rafcon.core.states.barrier_concurrency_state, 129
rafcon.core.states.concurrency_state, 131
rafcon.core.states.container_state, 132
rafcon.core.states.execution_state, 140
rafcon.core.states.hierarchy_state, 140
rafcon.core.states.library_state, 141
rafcon.core.states.preemptive_concurrency_state, 144
rafcon.core.states.state, 145
rafcon.core.storage.storage, 153
rafcon.gui.action, 262
rafcon.gui.clipboard, 267
rafcon.gui.controllers.execution_history, 178
rafcon.gui.controllers.global_variable_manager, 219
rafcon.gui.controllers.library_tree, 183
rafcon.gui.controllers.logging_console, 183
rafcon.gui.controllers.menu_bar, 185
rafcon.gui.controllers.modification_history, 189
rafcon.gui.controllers.state_editor.data_flows, 168
rafcon.gui.controllers.state_editor.description, 170
rafcon.gui.controllers.state_editor.linkage_overview, 165
rafcon.gui.controllers.state_editor.outcomes, 171
rafcon.gui.controllers.state_editor.scoped_variables, 165
rafcon.gui.controllers.state_editor.source_editor, 167
rafcon.gui.controllers.state_editor.state_editor, 170
rafcon.gui.controllers.state_editor.transitions, 173
rafcon.gui.controllers.state_icons, 196
rafcon.gui.controllers.state_machine_tree, 190
rafcon.gui.controllers.states_editor, 192
rafcon.gui.controllers.tool_bar, 195
rafcon.gui.controllers.top_tool_bar, 195
rafcon.gui.controllers.undocked_window, 197
rafcon.gui.controllers.utils.editor, 175
rafcon.gui.controllers.utils.extended_controller, 176
rafcon.gui.controllers.utils.single_widget_window, 178
rafcon.gui.helpers.label, 231
rafcon.gui.helpers.state, 234
rafcon.gui.helpers.state_machine, 237
rafcon.gui.helpers.text_formatting, 243
rafcon.gui.models.abstract_state, 198
rafcon.gui.models.auto_backup, 221
rafcon.gui.models.container_state, 204
rafcon.gui.models.data_flow, 209
rafcon.gui.models.data_port, 210
rafcon.gui.models.global_variable_manager, 223
rafcon.gui.models.library_manager, 223
rafcon.gui.models.library_state, 207
rafcon.gui.models.logical_port, 215
rafcon.gui.models.modification_history, 223
rafcon.gui.models.scoped_variable, 211
rafcon.gui.models.selection, 211
rafcon.gui.models.state, 202
rafcon.gui.models.state_element, 208
rafcon.gui.models.state_machine, 216
rafcon.gui.models.state_machine_execution_engine, 223
rafcon.gui.models.state_machine_manager, 223
rafcon.gui.mygaphas.canvas, 256
rafcon.gui.mygaphas.connector, 257
rafcon.gui.mygaphas.constraint, 258
rafcon.gui.mygaphas.guide, 259
rafcon.gui.mygaphas.items.connection, 245
rafcon.gui.mygaphas.items.line, 244
rafcon.gui.mygaphas.items.ports, 246
rafcon.gui.mygaphas.items.state, 248
rafcon.gui.mygaphas.segment, 260
rafcon.gui.mygaphas.utils.enums, 251
rafcon.gui.mygaphas.utils.gap_draw_helper, 253
rafcon.gui.mygaphas.utils.gap_helper, 253
rafcon.gui.mygaphas.view, 260
rafcon.gui.shortcut_manager, 269
rafcon.gui.singleton, 270
rafcon.gui.views.execution_history, 228
rafcon.gui.views.global_variable_editor, 228
rafcon.gui.views.library_tree, 228
rafcon.gui.views.logging_console, 228
rafcon.gui.views.main_window, 229

rafcon.gui.views.menu_bar, 230
 rafcon.gui.views.modification_history,
 230
 rafcon.gui.views.state_editor.data_flows,
 225
 rafcon.gui.views.state_editor.description_editorsource_editor, 167
 226
 rafcon.gui.views.state_editor.input_port_list, state_data_flows, 168
 225
 rafcon.gui.views.state_editor.linkage_overviewstate_element, 127
 224
 rafcon.gui.views.state_editor.outcomes,
 226
 rafcon.gui.views.state_editor.output_port_liststate_machine_status, 117
 225
 rafcon.gui.views.state_editor.overview,
 226
 rafcon.gui.views.state_editor.scoped_variableslist_transitions, 173
 225
 rafcon.gui.views.state_editor.source_editor, states_editor, 192
 225
 rafcon.gui.views.state_editor.state_editor,
 224
 rafcon.gui.views.state_editor.transitions,
 226
 rafcon.gui.views.state_machine_tree, 230
 rafcon.gui.views.state_machines_editor,
 231
 rafcon.gui.views.states_editor, 231
 rafcon.gui.views.tool_bar, 231
 rafcon.gui.views.utils.about_dialog, 227
 rafcon.gui.views.utils.editor, 227
 rafcon.gui.views.utils.single_widget_window,
 228
 rafcon.utils.constants, 271
 rafcon.utils.decorators, 271
 rafcon.utils.dict_operations, 271
 rafcon.utils.execution_log, 271
 rafcon.utils.filesystem, 272
 rafcon.utils.geometry, 273
 rafcon.utils.hashable, 274
 rafcon.utils.i18n, 275
 rafcon.utils.installation, 275
 rafcon.utils.log, 275
 rafcon.utils.log_helpers, 276
 rafcon.utils.multi_event, 277
 rafcon.utils.plugins, 278
 rafcon.utils.profiler, 278
 rafcon.utils.resources, 278
 rafcon.utils.storage_utils, 279
 rafcon.utils.timer, 279
 rafcon.utils.type_helpers, 279
 rafcon.utils.vividict, 280
 resources, 278
 scoped_variable, 124
 scoped_variable_list, 165
 script, 161
 single_widget_window, 178
 singleton, 162
 source_editor, 167
 state, 145
 state_editor, 170
 state_element, 127
 state_icons, 196
 state_machine, 162
 state_machine_manager, 164
 state_machine_status, 117
 state_machine_tree, 190
 state_outcomes, 171
 state_overview, 172
 state_transitions, 173
 states_editor, 192
 storage, 153
 storage_utils, 279
 tool_bar, 195
 top_tool_bar, 195
 transition, 126
 type_helpers, 279
 undocked_window, 197
 vividict, 280
 mouse_click() (rafcon.gui.controllers.execution_history.ExecutionHistory method), 180
 mouse_click() (rafcon.gui.controllers.library_tree.LibraryTreeController method), 184
 mouse_click() (rafcon.gui.controllers.state_machine_tree.StateMachineTree method), 190
 move() (rafcon.gui.myaphas.guide.GuidedNameInMotion method), 259
 move() (rafcon.gui.myaphas.guide.GuidedStateHandleInMotion method), 259
 move() (rafcon.gui.myaphas.guide.GuidedStateInMotion method), 259
 move_dirty_lock_file() (in module rafcon.gui.models.auto_backup), 222
 moving (rafcon.gui.myaphas.items.state.StateView property), 250
 multi_event
 module, 277
 mutable_hash() (rafcon.utils.hashable.Hashable method), 274
 MyCanvas (class in rafcon.gui.myaphas.canvas), 256

N

name (rafcon.core.state_elements.data_port.DataPort property), 122
 name (rafcon.core.state_elements.logical_port.Outcome property), 124

name (rafcon.core.state_elements.scope.ScopedData property), 125	notify_is_start() (rafcon.gui.controllers.state_editor.overview.StateOverviewController method), 173
name (rafcon.core.states.execution_state.ExecutionState property), 140	notify_name_change() (rafcon.gui.controllers.state_editor.overview.StateOverviewController method), 173
name (rafcon.core.states.state.State property), 150	notify_state_name_change() (rafcon.gui.controllers.states_editor.StatesEditorController method), 193
name (rafcon.gui.mygaphas.items.line.PerpLine property), 244	
name (rafcon.gui.mygaphas.items.ports.DataPortView property), 246	
name (rafcon.gui.mygaphas.items.ports.OutcomeView property), 247	O
name (rafcon.gui.mygaphas.items.ports.PortView property), 247	ObservableConfig (class in rafcon.core.config), 156
name (rafcon.gui.mygaphas.items.ports.ScopedVariablePortView property), 248	observe_model() (rafcon.gui.controllers.utils.extended_controller.ExtendedController method), 177
name (rafcon.gui.mygaphas.items.state.NameView property), 248	old_next_ids (rafcon.gui.models.modification_history.HistoryTreeElement property), 219
NAME_STORAGE_ID (rafcon.gui.controllers.global_variable_manager.GlobalVariableManager attribute), 181	on_about_activate() (rafcon.gui.controllers.menu_bar.MenuBarController static method), 186
NAME_STORAGE_ID (rafcon.gui.controllers.state_editor.outcomes.StateOutcomesList attribute), 171	on_add() (rafcon.gui.controllers.global_variable_manager.GlobalVariableManagerController), 182
NAME_STORAGE_ID (rafcon.gui.controllers.state_editor.scoped_variable_list.ScopedVariableList attribute), 166	on_add() (rafcon.gui.controllers.state_editor.data_flows.StateDataFlowsList attribute), 169
NAME_STORAGE_ID (rafcon.gui.controllers.state_machine_tree.StateMachineTreeController attribute), 190	on_add() (rafcon.gui.controllers.state_editor.scoped_variable_list.ScopedVariableListController), 171
name_view (rafcon.gui.mygaphas.items.state.StateView property), 250	on_add() (rafcon.gui.controllers.state_editor.transitions.StateTransitionsList attribute), 174
NameView (class in rafcon.gui.mygaphas.items.state), 248	on_add_state_activate() (rafcon.gui.controllers.menu_bar.MenuBarController static method), 186
negative_check_for_model_in_expected_future_models() (in module rafcon.gui.helpers.state), 235	on_add_transitions_from_closest_sibling_state_active() (rafcon.gui.controllers.menu_bar.MenuBarController static method), 186
new_buffer() (rafcon.gui.views.utils.editor.EditorView method), 227	on_add_transitions_to_closest_sibling_state_active() (rafcon.gui.controllers.menu_bar.MenuBarController static method), 186
new_change() (rafcon.gui.controllers.modification_history.ModificationHistoryTreeController method), 189	on_add_transitions_to_parent_state_active() (rafcon.gui.controllers.menu_bar.MenuBarController static method), 186
new_state_machine() (in module rafcon.gui.helpers.state_machine), 239	on_backward_step_activate() (rafcon.gui.controllers.menu_bar.MenuBarController static method), 186
next (rafcon.core.execution.execution_history_items.HistoryItem property), 116	on_bake_state_machine_activate() (rafcon.gui.controllers.menu_bar.MenuBarController static method), 186
next() (rafcon.gui.mygaphas.utils.enums.SnappedSide method), 251	on_button_bake_state_machine_clicked() (rafcon.gui.controllers.execution_history.ExecutionHistoryTreeController tool_bar.ToolBarController method), 195
next_id (rafcon.gui.models.modification_history.HistoryTreeElement property), 219	on_button_layout_state_machine() (rafcon.gui.controllers.execution_history.ExecutionHistoryTreeController tool_bar.ToolBarController method), 195
NoHigherLevelFilter (class in rafcon.utils.log_helpers), 277	
notification_selected_sm_changed() (rafcon.gui.controllers.execution_history.ExecutionHistoryTreeController method), 180	
notification_sm_changed() (rafcon.gui.controllers.execution_history.ExecutionHistoryTreeController method), 180	

on_button_new_clicked()	(raf- con.gui.controllers.tool_bar.ToolBarController method), 195	on_cursor_changed()	(raf- con.gui.controllers.modification_history.ModificationHistoryTree method), 189
on_button_open_clicked()	(raf- con.gui.controllers.tool_bar.ToolBarController method), 195	on_cut_selection_activate()	(raf- con.gui.controllers.menu_bar.MenuBarController method), 186
on_button_refresh_clicked()	(raf- con.gui.controllers.tool_bar.ToolBarController method), 195	on_data_flow_mode_toggled()	(raf- con.gui.controllers.menu_bar.MenuBarController static method), 186
on_button_refresh_libs_clicked()	(raf- con.gui.controllers.tool_bar.ToolBarController method), 195	on_delete_activate()	(raf- con.gui.controllers.menu_bar.MenuBarController method), 186
on_button_refresh_selected_clicked()	(raf- con.gui.controllers.tool_bar.ToolBarController method), 195	on_delete_check_sm_modified()	(raf- con.gui.controllers.menu_bar.MenuBarController method), 186
on_button_save_clicked()	(raf- con.gui.controllers.tool_bar.ToolBarController method), 195	on_delete_check_sm_running()	(raf- con.gui.controllers.menu_bar.MenuBarController method), 186
on_close_clicked()	(raf- con.gui.controllers.states_editor.StatesEditorController method), 193	on_delete_event()	(raf- con.gui.controllers.menu_bar.MenuBarController method), 186
on_combo_changed_from_key()	(raf- con.gui.controllers.state_editor.data_flows.StateDataFlowsListHolder method), 169	on_destroy()	(rafcon.gui.controllers.menu_bar.MenuBarController ListHolder)
on_combo_changed_from_outcome()	(raf- con.gui.controllers.state_editor.transitions.StateTransitionsListHolder method), 175	on_drag_begin()	(raf- con.gui.controllers.library_tree.LibraryTreeController ListHolder)
on_combo_changed_from_state()	(raf- con.gui.controllers.state_editor.data_flows.StateDataFlowsListHolder method), 169	on_drag_begin()	(raf- con.gui.controllers.state_icons.StateIconController ListHolder)
on_combo_changed_from_state()	(raf- con.gui.controllers.state_editor.transitions.StateTransitionsListHolder method), 175	on_drag_data_get()	(raf- con.gui.controllers.library_tree.LibraryTreeController ListHolder)
on_combo_changed_to_key()	(raf- con.gui.controllers.state_editor.data_flows.StateDataFlowsListHolder method), 169	on_drag_end()	(rafcon.gui.controllers.state_icons.StateIconController method), 196
on_combo_changed_to_outcome()	(raf- con.gui.controllers.state_editor.transitions.StateTransitionsListHolder method), 175	on_drag_end()	(rafcon.gui.controllers.state_editor.source_editor.SourceEditorView method), 225
on_combo_changed_to_state()	(raf- con.gui.controllers.state_editor.data_flows.StateDataFlowsListHolder method), 169	on_escape_key_press_event_leave_full_screen()	
on_combo_changed_to_state()	(raf- con.gui.controllers.state_editor.transitions.StateTransitionsListHolder method), 175	on_expert_view_activate()	(raf- con.gui.controllers.menu_bar.MenuBarController method), 187
on_config_value_changed()	(raf- con.gui.controllers.menu_bar.MenuBarController method), 186	on_focus()	(rafcon.gui.controllers.state_editor.data_flows.StateDataFlow method), 169
on_config_value_changed()	(raf- con.gui.controllers.states_editor.StatesEditorController method), 193	on_focus()	(rafcon.gui.controllers.state_editor.transitions.StateTransition method), 175
on_copy_selection_activate()	(raf- con.gui.controllers.menu_bar.MenuBarController method), 186	on_focus_out()	(raf- con.gui.controllers.state_editor.description_editor.DescriptionEd method), 170
		on_focus_out()	(raf- con.gui.controllers.state_editor.overview.StateOverviewControlle method), 170

method), 173
 on_full_screen_activate() (raf- con.gui.controllers.menu_bar.MenuBarController method), 187
 on_full_screen_deactivate() (raf- con.gui.controllers.menu_bar.MenuBarController method), 187
 on_full_screen_mode_toggled() (raf- con.gui.controllers.menu_bar.MenuBarController method), 187
 on_grid_toggled() (raf- con.gui.controllers.menu_bar.MenuBarController method), 187
 on_group_states_activate() (raf- con.gui.controllers.menu_bar.MenuBarController method), 187
 on_layout_state_machine() (raf- con.gui.controllers.menu_bar.MenuBarController static method), 187
 on_lock() (rafcon.gui.controllers.global_variable_manager.GlobalVariableManagerMenuBarController method), 182
 on_maximize_button_clicked() (raf- con.gui.controllers.top_tool_bar.TopToolBarController method), 196
 on_menu_preferences_activate() (raf- con.gui.controllers.menu_bar.MenuBarController static method), 187
 on_minimize_button_clicked() (raf- con.gui.controllers.top_tool_bar.TopToolBarController method), 196
 on_model_destruct() (raf- con.gui.models.selection.Selection method), 213
 on_mouse_click() (raf- con.gui.controllers.state_icons.StateIconController method), 197
 on_mouse_motion() (raf- con.gui.controllers.state_icons.StateIconController method), 197
 on_new_activate() (raf- con.gui.controllers.menu_bar.MenuBarController method), 187
 on_open_activate() (raf- con.gui.controllers.menu_bar.MenuBarController static method), 187
 on_open_library_state_separately_activate() (rafcon.gui.controllers.menu_bar.MenuBarController static method), 187
 on_paste_clipboard_activate() (raf- con.gui.controllers.menu_bar.MenuBarController method), 187
 on_pause_activate() (raf- con.gui.controllers.menu_bar.MenuBarController method), 187

on_quit_activate() (raf- con.gui.controllers.menu_bar.MenuBarController method), 187
 on_redo_activate() (raf- con.gui.controllers.menu_bar.MenuBarController method), 187
 on_redo_button_clicked() (raf- con.gui.controllers.modification_history.ModificationHistoryTree method), 189
 on_redock_button_clicked() (raf- con.gui.controllers.top_tool_bar.TopToolBarUndockedWindowController method), 196
 on_refresh_all_activate() (raf- con.gui.controllers.menu_bar.MenuBarController method), 187
 on_refresh_libraries_activate() (raf- con.gui.controllers.menu_bar.MenuBarController static method), 187
 on_refresh_selected_activate() (raf- con.gui.controllers.menu_bar.MenuBarController static method), 187
 on_reset_button_clicked() (raf- con.gui.controllers.modification_history.ModificationHistoryTree method), 189
 on_right_click_menu() (raf- con.gui.controllers.state_editor.data_flows.StateDataFlowsListController method), 169
 on_right_click_menu() (raf- con.gui.controllers.state_editor.outcomes.StateOutcomesListController method), 172
 on_right_click_menu() (raf- con.gui.controllers.state_editor.scoped_variable_list.ScopedVariableListController method), 166
 on_right_click_menu() (raf- con.gui.controllers.state_editor.transitions.StateTransitionsListController method), 175
 on_run_only_selected_state_activate() (raf- con.gui.controllers.menu_bar.MenuBarController method), 187
 on_run_selected_state_activate() (raf- con.gui.controllers.menu_bar.MenuBarController method), 187
 on_run_to_selected_state_activate() (raf- con.gui.controllers.menu_bar.MenuBarController method), 187
 on_save_activate() (raf- con.gui.controllers.menu_bar.MenuBarController method), 187
 on_save_as_activate() (raf- con.gui.controllers.menu_bar.MenuBarController method), 187
 on_save_as_copy_activate() (raf- con.gui.controllers.menu_bar.MenuBarController method), 187

```

on_save_selected_state_as_activate()      (raf- on_to_outcome_edited())          (raf-
con.gui.controllers.menu_bar.MenuBarController           con.gui.controllers.state_editor.outcomes.StateOutcomesListCont
static method), 187                      method), 172

on_search_activate()                     (raf- on_to_state_edited())          (raf-
con.gui.controllers.menu_bar.MenuBarController           con.gui.controllers.state_editor.outcomes.StateOutcomesListCont
method), 187                           method), 172

on_show_aborted_preempted_toggled()     (raf- on_toggle_full_screen_mode())    (raf-
con.gui.controllers.menu_bar.MenuBarController           con.gui.controllers.menu_bar.MenuBarController
static method), 187                      method), 188

on_show_data_flows_toggled()           (raf- on_toggle_is_start_state())       (raf-
con.gui.controllers.menu_bar.MenuBarController           con.gui.controllers.state_editor.overview.StateOverviewContro
static method), 187                      method), 173

on_show_data_values_toggled()          (raf- on_toggle_is_start_state_active()) (raf-
con.gui.controllers.menu_bar.MenuBarController           con.gui.controllers.menu_bar.MenuBarController
static method), 188                      static method), 188

on_show_transitions_toggled()          (raf- on_toggle_mode())                 (raf-
con.gui.controllers.menu_bar.MenuBarController           con.gui.controllers.modification_history.ModificationHistoryTree
static method), 188                      method), 189

on_start_activate()                   (raf- on_toggle_show_content())        (raf-
con.gui.controllers.menu_bar.MenuBarController           con.gui.controllers.state_editor.overview.StateOverviewContro
method), 188                           method), 173

on_start_from_selected_state_activate() (raf- on_toggle_sticky_clicked())       (raf-
con.gui.controllers.menu_bar.MenuBarController           con.gui.controllers.states_editor.StatesEditorController
method), 188                           method), 193

on_step_into_activate()               (raf- on_toggle_tree_folded())         (raf-
con.gui.controllers.menu_bar.MenuBarController           con.gui.controllers.modification_history.ModificationHistoryTree
method), 188                           method), 189

on_step_mode_activate()              (raf- on_undo_activate())             (raf-
con.gui.controllers.menu_bar.MenuBarController           con.gui.controllers.menu_bar.MenuBarController
method), 188                           method), 188

on_step_out_activate()               (raf- on_undo_button_clicked())        (raf-
con.gui.controllers.menu_bar.MenuBarController           con.gui.controllers.modification_history.ModificationHistoryTree
method), 188                           method), 189

on_step_over_activate()              (raf- on_ungroup_state_activate())     (raf-
con.gui.controllers.menu_bar.MenuBarController           con.gui.controllers.menu_bar.MenuBarController
method), 188                           method), 188

on_stop_activate()                  (raf- on_unlock()) (rafcon.gui.controllers.global_variable_manager.GlobalVari
con.gui.controllers.menu_bar.MenuBarController           method), 182

on_substitute_library_with_template_activate() (raf- ongoing_complex_actions)      (raf-
rafcon.gui.controllers.menu_bar.MenuBarController           con.gui.models.state_machine.ComplexActionObserver
static method), 188                      property), 216

on_substitute_selected_state_activate() (raf- ongoing_complex_actions)      (raf-
con.gui.controllers.menu_bar.MenuBarController           con.gui.models.state_machine.StateMachineModel
static method), 188                      property), 218

on_switch_page()                   (raf- open_button_clicked())            (raf-
con.gui.controllers.states_editor.StatesEditorController   con.gui.controllers.library_tree.LibraryTreeController
method), 193                           method), 185

on_tab_close_clicked()              (raf- open_folder_cmd_line())          (in module raf-
con.gui.controllers.states_editor.StatesEditorController   con.core.interface), 160
method), 193                           160

on_text_view_event()               (raf- open_folder_func()) (in module rafcon.core.interface),
con.gui.views.state_editor.SourceEditorView           con.gui.controllers.library_tree.LibraryTreeController
method), 225                           method), 185

```

open_library_state_separately() (in module `rafcon.gui.helpers.state_machine`), 239
open_run_button_clicked() (rafcon.gui.controllers.library_tree.LibraryTreeController method), 185
open_selected_history_separately() (rafcon.gui.controllers.execution_history.ExecutionHistoryTree method), 180
open_state_machine() (in module `rafcon.gui.helpers.state_machine`), 239
opposite() (rafcon.gui.mygaphas.utils.enums.SnappedSide method), 251
or_clear() (in module `rafcon.utils.multi_event`), 277
or_set() (in module `rafcon.utils.multi_event`), 277
orify() (in module `rafcon.utils.multi_event`), 277
OS_PATH_STORAGE_ID (rafcon.gui.controllers.library_tree.LibraryTreeController attribute), 183
outcome module, 123
Outcome (class in `rafcon.core.state_elements.logical_port`), 123
outcome (rafcon.gui.models.logical_port.OutcomeModel property), 216
outcome_id (rafcon.core.state_elements.logical_port.Outcome property), 124
outcome_id (rafcon.gui.mygaphas.items.ports.OutcomeView property), 247
outcome_port() (rafcon.gui.mygaphas.items.state.StateView method), 250
OutcomeAction (class in `rafcon.gui.action`), 264
OutcomeModel (class in `rafcon.gui.models.logical_port`), 215
outcomes (`rafcon.core.states.state.State` property), 150
outcomes (rafcon.gui.models.abstract_state.AbstractStateModel property), 201
outcomes (rafcon.gui.models.selection.Selection property), 213
outcomes (rafcon.gui.mygaphas.items.state.StateView property), 250
outcomes_changed() (rafcon.gui.controllers.state_editor.outcomes.StateOutcomesListController method), 172
OutcomeView (class in `rafcon.gui.mygaphas.items.ports`), 246
output_data (`rafcon.core.states.state.State` property), 150
output_data_port_runtime_values (rafcon.core.states.library_state.LibraryState property), 143
output_data_ports (`rafcon.core.states.state.State` property), 150
output_data_ports (rafcon.gui.models.abstract_state.AbstractStateModel property), 201
output_data_ports (rafcon.gui.models.selection.Selection property), 213
output_port() (rafcon.gui.mygaphas.items.state.StateView method), 250
OutputDataPort (class in `rafcon.core.state_elements.data_port`), 122
OutputPortsListView (class in `rafcon.gui.views.state_editor.output_port_list`), 225
OutputPortView (class in `rafcon.gui.mygaphas.items.ports`), 247
outputs (rafcon.gui.mygaphas.items.state.StateView property), 250

P

pane_position_check() (rafcon.gui.views.state_editor.source_editor.SourceEditorView method), 225
parent (rafcon.core.script.Script property), 161
parent (rafcon.core.state_elements.state_element.StateElement property), 128
parent (rafcon.core.states.state.State property), 150
parent (rafcon.gui.controllers.utils.extended_controller.ExtendedController property), 177
parent (rafcon.gui.models.abstract_state.AbstractStateModel property), 201
parent (rafcon.gui.models.state_element.StateElementModel property), 208
parent (rafcon.gui.mygaphas.items.line.PerpLine property), 244
parent (rafcon.gui.mygaphas.items.ports.PortView property), 247
parent (rafcon.gui.mygaphas.items.state.NameView property), 249
parent (rafcon.gui.mygaphas.items.state.StateView property), 250
paste() (rafcon.gui.clipboard.Clipboard method), 268
paste_action_callback() (rafcon.gui.controllers.state_editor.outcomes.StateOutcomesEditorController method), 171
paste_action_callback() (rafcon.gui.controllers.state_editor.scoped_variable_list.ScopedVariableListController method), 166
paste_action_callback() (rafcon.gui.controllers.state_machine.StateMachineTreeController method), 190
paste_into_selected_state() (in module `rafcon.gui.helpers.state_machine`), 240
path (rafcon.core.script.Script property), 161
paused (`rafcon.core.states.state.State` property), 150

perform_temp_storage()	(raf-con.gui.models.auto_backup.AutoBackupModel method), 222	possible_method_names	(raf-con.gui.action.DataPortAction attribute), 264
PerpLine (class in rafcon.gui.mygaphas.items.line), 244		possible_method_names	(raf-con.gui.action.OutcomeAction attribute), 264
plugins module, 278		possible_method_names	(raf-con.gui.action.RemoveObjectAction attribute), 265
point() (rafcon.gui.mygaphas.items.line.PerpLine method), 244		possible_method_names	(raf-con.gui.action.StateAction attribute), 265
point_left_of_line() (in module raf-con.utils.geometry), 274		possible_method_names	(raf-con.gui.action.StateElementAction attribute), 266
pop_last_item()	(raf-con.core.execution.in_memory_execution_history method), 117	possible_method_names	(raf-con.gui.action.TransitionAction attribute), 267
port_id(rafcon.gui.mygaphas.items.ports.DataPortView property), 246		preempted (rafcon.core.states.state.State property), 150	
port_id(rafcon.gui.mygaphas.items.ports.ScopedVariablePortView property), 248		preemptive_concurrency_state module, 144	
port_side_size	(raf-con.gui.mygaphas.items.ports.PortView property), 247	preemptive_wait()	(rafcon.core.states.state.State method), 150
port_size (rafcon.gui.mygaphas.items.ports.PortView property), 247		PreemptiveConcurrencyState (class in raf-con.core.states.preemptive_concurrency_state), 144	
port_size(rafcon.gui.mygaphas.items.ports.ScopedVariablePortView property), 248		prepare_destruction()	(raf-con.gui.action.AbstractAction method), 262
PortRectConstraint (class in raf-con.gui.mygaphas.constraint), 258		prepare_destruction()	(raf-con.gui.controllers.states_editor.StatesEditorController method), 193
PortView (class in rafcon.gui.mygaphas.items.ports), 247		prepare_destruction()	(raf-con.gui.models.abstract_state.AbstractStateModel method), 201
pos (rafcon.gui.mygaphas.canvas.ItemProjection property), 256		prepare_destruction()	(raf-con.gui.models.auto_backup.AutoBackupModel method), 222
pos (rafcon.gui.mygaphas.items.ports.PortView property), 247		prepare_destruction()	(raf-con.gui.models.container_state.ContainerStateModel method), 205
position (rafcon.gui.mygaphas.items.state.NameView property), 249		prepare_destruction()	(raf-con.gui.models.data_flow.DataFlowModel method), 210
position (rafcon.gui.mygaphas.items.state.StateView property), 250		prepare_destruction()	(raf-con.gui.models.data_port.DataPortModel method), 210
possible_args (rafcon.gui.action.DataFlowAction attribute), 263		prepare_destruction()	(raf-con.gui.models.library_state.LibraryStateModel method), 207
possible_args (rafcon.gui.action.DataPortAction attribute), 264		prepare_destruction()	(raf-con.gui.models.logical_port.OutcomeModel method), 216
possible_args (rafcon.gui.action.OutcomeAction attribute), 264		prepare_destruction()	(raf-con.gui.models.modification_history.HistoryTreeElement method), 219
possible_args (rafcon.gui.action.StateAction attribute), 265			
possible_args (rafcon.gui.action.StateElementAction attribute), 266			
possible_args (rafcon.gui.action.TransitionAction attribute), 267			
possible_method_names	(raf-con.gui.action.AddObjectAction attribute), 263		
possible_method_names	(raf-con.gui.action.DataFlowAction attribute), 264		

prepare_destruction() (raf-
con.gui.models.modification_history.ModificationHistoryItem) (raf-
method), 220
con.core.execution.in_memory_execution_history.InMemoryExecutionHistory

prepare_destruction() (raf-
con.gui.models.modification_history.ModificationHistoryReturnHistoryItem) (raf-
method), 221
con.core.execution.base_execution_history.BaseExecutionHistory

prepare_destruction() (raf-
con.gui.models.scoped_variable.ScopedVariableModel.push_return_history_item) (raf-
method), 211
con.core.execution.in_memory_execution_history.InMemoryExecutionHistory

prepare_destruction() (raf-
con.gui.models.state_element.StateElementModel.push_state_machine_start_history_item) (raf-
method), 208
con.core.execution.base_execution_history.BaseExecutionHistory

prepare_destruction() (raf-
con.gui.models.state_machine.StateMachineMode.push_state_machine_start_history_item) (raf-
method), 218
con.core.execution.in_memory_execution_history.InMemoryExecutionHistory

prepare_destruction() (raf-
con.gui.models.transition.TransitionModel) Python Enhancement Proposals
method), 209 PEP 287, 104

prepare_destruction() (raf-
con.gui.mygaphas.view.ExtendedGtkView) PEP 8, 104
method), 261

prepare_new_copy() (rafcon.gui.clipboard.Clipboard) queue_draw_item() (raf-
method), 269
con.gui.mygaphas.view.ExtendedGtkView

prepare_state_m_for_insert_as() (in module raf-
con.gui.helpers.state), 236
method), 261

prepare_the_labels() (raf-
con.gui.views.state_editor.state_editor.StateEditorView) R
method), 224 rafcon.core.config
module, 156

prev(rafcon.core.execution.execution_history_items.HistoryItem) rafcon.core.constants
property), 116 module, 158

prev() (rafcon.gui.mygaphas.utils.enums.SnappedSide) rafcon.core.custom_exceptions
method), 251 module, 157

prev_id(rafcon.gui.models.modification_history.HistoryTreeElement) rafcon.core.execution.base_execution_history
property), 219 module, 112

print_filtered_buffer() (raf- rafcon.core.execution.consumer_manager
con.gui.controllers.logging_console.LoggingConsoleController) module, 114
method), 183 rafcon.core.execution.consumers.abstract_execution_history
module, 119

print_message() (raf- rafcon.core.execution.consumers.file_system_consumer
con.gui.controllers.logging_console.LoggingConsoleController) module, 119
method), 183 rafcon.core.execution.execution_engine
module, 115

print_message() (raf- rafcon.core.execution.execution_history_factory
con.gui.views.logging_console.LoggingConsoleView) module, 115
method), 229 rafcon.core.execution.execution_history_items
module, 115

print_to_text_view() (raf- rafcon.core.execution.execution_status
con.gui.views.logging_console.LoggingConsoleView) module, 117
method), 229 rafcon.core.execution.in_memory_execution_history
module, 117

push_call_history_item() (raf- rafcon.core.execution.in_memory_execution_history
con.core.execution.base_execution_history.BaseExecutionHistory) module, 117
method), 112 rafcon.core.global_variable_manager
module, 158

push_call_history_item() (raf- rafcon.core.id_generator
con.core.execution.in_memory_execution_history.InMemoryExecutionHistory) module, 158
method), 117

push_concurrency_history_item() (raf- rafcon.core.id_generator
con.core.execution.base_execution_history.BaseExecutionHistory) module, 158
method), 117

Q

R

rafcon.core.interface	rafcon.gui.controllers.menu_bar
module, 159	module, 185
rafcon.core.library_manager	rafcon.gui.controllers.modification_history
module, 161	module, 189
rafcon.core.script	rafcon.gui.controllers.state_editor.data_flows
module, 161	module, 168
rafcon.core.singleton	rafcon.gui.controllers.state_editor.description_editor
module, 162	module, 170
rafcon.core.state_elements.data_flow	rafcon.gui.controllers.state_editor.linkage_overview
module, 120	module, 165
rafcon.core.state_elements.data_port	rafcon.gui.controllers.state_editor.outcomes
module, 121	module, 171
rafcon.core.state_elements.logical_port	rafcon.gui.controllers.state_editor.overview
module, 123	module, 172
rafcon.core.state_elements.scope	rafcon.gui.controllers.state_editor.scoped_variable_list
module, 124	module, 165
rafcon.core.state_elements.state_element	rafcon.gui.controllers.state_editor.source_editor
module, 127	module, 167
rafcon.core.state_elements.transition	rafcon.gui.controllers.state_editor.state_editor
module, 126	module, 170
rafcon.core.state_machine	rafcon.gui.controllers.state_editor.transitions
module, 162	module, 173
rafcon.core.state_machine_manager	rafcon.gui.controllers.state_icons
module, 164	module, 196
rafcon.core.states.barrier_concurrency_state	rafcon.gui.controllers.state_machine_tree
module, 129	module, 190
rafcon.core.states.concurrency_state	rafcon.gui.controllers.states_editor
module, 131	module, 192
rafcon.core.states.container_state	rafcon.gui.controllers.tool_bar
module, 132	module, 195
rafcon.core.states.execution_state	rafcon.gui.controllers.top_tool_bar
module, 140	module, 195
rafcon.core.states.hierarchy_state	rafcon.gui.controllers.undocked_window
module, 140	module, 197
rafcon.core.states.library_state	rafcon.gui.controllers.utils.editor
module, 141	module, 175
rafcon.core.states.preemptive_concurrency_state	rafcon.gui.controllers.utils.extended_controller
module, 144	module, 176
rafcon.core.states.state	rafcon.gui.controllers.utils.single_widget_window
module, 145	module, 178
rafcon.core.storage.storage	rafcon.gui.helpers.label
module, 153	module, 231
rafcon.gui.action	rafcon.gui.helpers.state
module, 262	module, 234
rafcon.gui.clipboard	rafcon.gui.helpers.state_machine
module, 267	module, 237
rafcon.gui.controllers.execution_history	rafcon.gui.helpers.text_formatting
module, 178	module, 243
rafcon.gui.controllers.global_variable_manager	rafcon.gui.models.abstract_state
module, 181	module, 198
rafcon.gui.controllers.library_tree	rafcon.gui.models.auto_backup
module, 183	module, 221
rafcon.gui.controllers.logging_console	rafcon.gui.models.container_state
module, 183	module, 204

rafcon.gui.models.data_flow
 module, 209
rafcon.gui.models.data_port
 module, 210
rafcon.gui.models.global_variable_manager
 module, 223
rafcon.gui.models.library_manager
 module, 223
rafcon.gui.models.library_state
 module, 207
rafcon.gui.models.logical_port
 module, 215
rafcon.gui.models.modification_history
 module, 219
rafcon.gui.models.scoped_variable
 module, 211
rafcon.gui.models.selection
 module, 211
rafcon.gui.models.state
 module, 202
rafcon.gui.models.state_element
 module, 208
rafcon.gui.models.state_machine
 module, 216
rafcon.gui.models.state_machine_execution_engine
 module, 223
rafcon.gui.models.state_machine_manager
 module, 223
rafcon.gui.models.transition
 module, 209
rafcon.gui.mygaphas.canvas
 module, 256
rafcon.gui.mygaphas.connector
 module, 257
rafcon.gui.mygaphas.constraint
 module, 258
rafcon.gui.mygaphas.guide
 module, 259
rafcon.gui.mygaphas.items.connection
 module, 245
rafcon.gui.mygaphas.items.line
 module, 244
rafcon.gui.mygaphas.items.ports
 module, 246
rafcon.gui.mygaphas.items.state
 module, 248
rafcon.gui.mygaphas.segment
 module, 260
rafcon.gui.mygaphas.utils.enums
 module, 251
rafcon.gui.mygaphas.utils.gap_draw_helper
 module, 252
rafcon.gui.mygaphas.utils.gap_helper
 module, 253

rafcon.gui.mygaphas.view
 module, 260
rafcon.gui.shortcut_manager
 module, 269
rafcon.gui.singleton
 module, 270
rafcon.gui.views.execution_history
 module, 228
rafcon.gui.views.global_variable_editor
 module, 228
rafcon.gui.views.library_tree
 module, 228
rafcon.gui.views.logging_console
 module, 228
rafcon.gui.views.main_window
 module, 229
rafcon.gui.views.menu_bar
 module, 230
rafcon.gui.views.modification_history
 module, 230
rafcon.gui.views.state_editor.data_flows
 module, 225
rafcon.gui.views.state_editor.description_editor
 module, 226
rafcon.gui.views.state_editor.input_port_list
 module, 225
rafcon.gui.views.state_editor.linkage_overview
 module, 224
rafcon.gui.views.state_editor.outcomes
 module, 226
rafcon.gui.views.state_editor.output_port_list
 module, 225
rafcon.gui.views.state_editor.overview
 module, 226
rafcon.gui.views.state_editor.scoped_variables_list
 module, 225
rafcon.gui.views.state_editor.source_editor
 module, 225
rafcon.gui.views.state_editor.state_editor
 module, 224
rafcon.gui.views.state_editor.transitions
 module, 226
rafcon.gui.views.state_machine_tree
 module, 230
rafcon.gui.views.state_machines_editor
 module, 231
rafcon.gui.views.states_editor
 module, 231
rafcon.gui.views.tool_bar
 module, 231
rafcon.gui.views.utils.about_dialog
 module, 227
rafcon.gui.views.utils.editor
 module, 227

rafcon.gui.views.utils.single_widget_window
 module, 228

rafcon.utils.constants
 module, 271

rafcon.utils.decorators
 module, 271

rafcon.utils.dict_operations
 module, 271

rafcon.utils.execution_log
 module, 271

rafcon.utils.filesystem
 module, 272

rafcon.utils.geometry
 module, 273

rafcon.utils.hashable
 module, 274

rafcon.utils.i18n
 module, 275

rafcon.utils.installation
 module, 275

rafcon.utils.log
 module, 275

rafcon.utils.log_helpers
 module, 276

rafcon.utils.multi_event
 module, 277

rafcon.utils.plugins
 module, 278

rafcon.utils.profiler
 module, 278

rafcon.utils.resources
 module, 278

rafcon.utils.storage_utils
 module, 279

rafcon.utils.timer
 module, 279

rafcon.utils.type_helpers
 module, 279

rafcon.utils.vividict
 module, 280

RAFCON_LIBRARY_PATH, 39, 59, 78

RAFCON_LOGGING_CONF, 30, 77, 78

RAFCON_PLUGIN_PATH, 78, 87, 89

RAFCON_START_MINIMIZED, 24, 78

RAFCONDeprecationWarning, 275

re_initiate_meta_data()
 (raf-
 con.gui.models.modification_history.ModificationsHistoryModel
 method), 221

re_initiate_model_list()
 (raf-
 con.gui.models.state.StateModel
 method), 203

react_to_event() (in module raf-
 con.gui.helpers.label), 233

read_file() (in module rafcon.utils.filesystem), 273

reconnect_data_flow() (in module raf-
 con.core.storage.storage), 155

recover_state_machine_from_backup() (in module rafcon.gui.models.auto_backup), 222

RecoveryModeException, 157

RectanglePointPort (class in raf-
 con.gui.mygaphas.connector), 257

recursive_generate_models() (raf-
 con.gui.models.library_state.LibraryStateModel
 method), 207

recursively_pause_states() (raf-
 con.core.states.container_state.ContainerState
 method), 137

recursively_pause_states() (raf-
 con.core.states.library_state.LibraryState
 method), 143

recursively_pause_states() (raf-
 con.core.states.state.State method), 151

recursively_preempt_states() (raf-
 con.core.states.container_state.ContainerState
 method), 137

recursively_preempt_states() (raf-
 con.core.states.library_state.LibraryState
 method), 143

recursively_preempt_states() (raf-
 con.core.states.state.State method), 151

recursively_resume_states() (raf-
 con.core.states.container_state.ContainerState
 method), 137

recursively_resume_states() (raf-
 con.core.states.library_state.LibraryState
 method), 143

recursively_resume_states() (raf-
 con.core.states.state.State method), 151

redo() (rafcon.gui.action.AbstractAction method), 262

redo() (rafcon.gui.action.Action method), 263

redo() (rafcon.gui.action.AddObjectAction method), 263

redo() (rafcon.gui.action.DataFlowAction method), 264

redo() (rafcon.gui.action.DataPortAction method), 264

redo() (rafcon.gui.action.MetaDataAction method), 264

redo() (rafcon.gui.action.OutcomeAction method), 264

redo() (rafcon.gui.action.RemoveObjectAction method), 265

redo() (rafcon.gui.action.StateAction method), 265

redo() (rafcon.gui.action.StateMachineAction method), 266

redo() (rafcon.gui.action.TransitionAction method), 267

redo() (rafcon.gui.controllers.modification_history.ModificationHistoryTr
 method), 189

redo() (rafcon.gui.models.modification_history.ModificationsHistory
 method), 220

redo() (rafcon.gui.models.modification_history.ModificationsHistoryMode
 method), 221

redo_expansion_state() (raf-
con.gui.controllers.library_tree.LibraryTreeController method), 173

register_actions() (raf-
con.gui.controllers.state_machine_tree.StateMachineTreeController method), 191

redo_expansion_state() (raf-
con.gui.controllers.state_machine_tree.StateMachineTreeController method), 190

register_actions() (raf-
con.gui.controllers.states_editor.StatesEditorController method), 193

reduce_to_parent_states() (in module raf-
con.gui.helpers.state_machine), 240

register_actions() (raf-
con.gui.controllers.tool_bar.ToolBarController method), 195

reduce_to_parent_states() (in module raf-
con.gui.models.selection), 214

refresh_after_relocate_and_rename() (in module raf-
con.gui.helpers.state_machine), 240

register_actions() (raf-
con.gui.controllers.utils.editor.EditorController method), 176

refresh_all() (in module raf-
con.gui.helpers.state_machine), 240

register_actions() (raf-
con.gui.controllers.utils.extended_controller.ExtendedController method), 177

refresh_libraries() (in module raf-
con.gui.helpers.state_machine), 240

register_consumer() (raf-
con.core.execution.consumer_manager.ExecutionHistoryConsumer method), 114

refresh_selected_state_machine() (in module raf-
con.gui.helpers.state_machine), 240

register_shortcuts() (raf-
con.gui.controllers.menu_bar.MenuBarController register_shortcuts() (raf-
method), 188

register() (rafcon.core.execution.consumers.abstract_execution_history.Consumer) (raf-
method), 119

register_view() (raf-
con.gui.shortcut_manager.ShortcutManager method), 269

register() (rafcon.core.execution.consumers.file_system_consumer.FileSystemController execution_history.ExecutionHistoryTreeController method), 119

register_view() (raf-
method), 180

register() (rafcon.gui.controllers.modification_history.ModificationHistoryTreeController) (raf-
method), 189

register_view() (raf-
con.gui.controllers.global_variable_manager.GlobalVariableManager method), 191

register() (rafcon.gui.controllers.state_machine_tree.StateMachineTreeController) (raf-
method), 191

register_view() (raf-
method), 191

register() (rafcon.gui.views.utils.editor.EditorView) (raf-
con.gui.controllers.library_tree.LibraryTreeController method), 185

register_actions() (raf-
con.gui.controllers.global_variable_manager.GlobalVariableManager) (raf-
method), 182

register_view() (raf-
con.gui.controllers.modification_history.ModificationHistoryTreeController) (raf-
method), 183

register_actions() (raf-
con.gui.controllers.menu_bar.MenuBarController) (raf-
method), 188

register_view() (raf-
con.gui.controllers.menu_bar.MenuBarController) (raf-
method), 188

register_actions() (raf-
con.gui.controllers.modification_history.ModificationHistoryTreeController) (raf-
method), 188

register_view() (raf-
con.gui.controllers.modification_history.ModificationHistoryTreeController) (raf-
method), 189

register_actions() (raf-
con.gui.controllers.state_editor.data_flows.StateDataFlowsEditor) (raf-
method), 168

register_view() (raf-
con.gui.controllers.state_editor.data_flows.StateDataFlowsEditor) (raf-
method), 168

register_actions() (raf-
con.gui.controllers.state_editor.description_editor.DescriptionEditor) (raf-
method), 170

register_view() (raf-
con.gui.controllers.state_editor.description_editor.DescriptionEditor) (raf-
method), 169

register_actions() (raf-
con.gui.controllers.state_editor.outcomes.StateOutcomesEditor) (raf-
method), 171

register_view() (raf-
con.gui.controllers.state_editor.outcomes.StateOutcomesEditor) (raf-
method), 170

register_actions() (raf-
con.gui.controllers.state_editor.scoped_variable_list.ScopedVariableListEditor) (raf-
method), 166

register_view() (raf-
con.gui.controllers.state_editor.outcomes.StateOutcomesEditor) (raf-
method), 171

register_actions() (raf-
con.gui.controllers.state_editor.transitions.StateTransitionsEditor) (raf-
method), 171

register_view() (raf-
con.gui.controllers.state_editor.outcomes.StateOutcomesEditor) (raf-
method), 171

method), 172
register_view() *(raf-* *method), 178*
con.gui.controllers.state_editor.overview.StateOverviewController *(raf-*
method), 173 *reload_history()* *(raf-*
method), 180
register_view() *(raf-* *method), 166* *reload_scoped_variables_list_store()* *(raf-*
con.gui.controllers.state_editor.scoped_variable_list.ScopedVariableListController *(raf-*
method), 167
register_view() *(raf-* *method), 167* *reload_style()* *(raf-*
con.gui.controllers.state_editor.source_editor.SourceEditorController *(raf-*
method), 193
register_view() *(raf-* *method), 170* *relocate_libraries()* *(in module raf-*
con.gui.controllers.state_editor.state_editor.StateEditorController *(raf-*
method), 240 *relocate_library()* *(in module raf-*
register_view() *(raf-* *method), 173* *relocate_libraries()* *(in module raf-*
con.gui.controllers.state_editor.transitions.StateTransitionEditorController *(raf-*
method), 241 *con.gui.helpers.state_machine), 240*
register_view() *(raf-* *method), 175* *remove()* *(rafcon.core.states.container_state.ContainerState*
con.gui.controllers.state_editor.transitions.StateTransitionsListBodyController *(raf-*
method), 151 *remove()* *(rafcon.core.states.state.State method), 151*
register_view() *(raf-* *method), 197* *remove()* *(rafcon.gui.models.selection.Selection*
con.gui.controllers.state_icons.StateIconController *(raf-*
method), 213 *remove()* *(rafcon.gui.mygaphas.canvas.MyCanvas*
register_view() *(raf-* *method), 191* *remove()* *(rafcon.gui.mygaphas.items.connection.ConnectionView*
con.gui.controllers.state_machine_tree.StateMachineTreeController *(raf-*
method), 245
register_view() *(raf-* *method), 193* *remove()* *(rafcon.gui.mygaphas.items.line.PerpLine*
con.gui.controllers.states_editor.StatesEditorController *(raf-*
method), 245 *remove()* *(rafcon.gui.mygaphas.items.state.NameView*
register_view() *(raf-* *method), 195* *remove()* *(rafcon.gui.mygaphas.items.state.StateView*
con.gui.controllers.tool_bar.ToolBarController *(raf-*
method), 250
register_view() *(raf-* *method), 196* *remove_additional_model()* *(raf-*
con.gui.controllers.top_tool_bar.TopToolBarController *(raf-*
method), 203 *con.gui.models.state.StateModel* *(raf-*
register_view() *(raf-* *method), 196* *remove_all_waypoints()* *(raf-*
con.gui.controllers.top_tool_bar.TopToolBarUndockedWindowController *(raf-*
method), 245 *con.gui.mygaphas.items.line.PerpLine* *(raf-*
register_view() *(raf-* *method), 176* *remove_callback_for_action()* *(raf-*
con.gui.controllers.utils.editor.EditorController *(raf-*
method), 269 *con.gui.shortcut_manager.ShortcutManager*
register_view() *(raf-* *method), 177* *remove_callbacks_for_controller()* *(raf-*
con.gui.controllers.utils.extended_controller.ExtendedController *(raf-*
method), 270 *con.gui.shortcut_manager.ShortcutManager*
register_view() *(raf-* *method), 178* *remove_connected_handle()* *(raf-*
con.gui.controllers.utils.single_widget_window.SingleWidgetWindowController *(raf-*
method), 247 *con.gui.mygaphas.items.ports.PortView*
release_modification_lock() *(raf-* *method), 163* *remove_connection_from_ports()* *(raf-*
con.core.state_machine.StateMachine *(raf-*
method), 245 *con.gui.mygaphas.items.connection.ConnectionView*
relieve_all_models() *(raf-* *method), 177* *remove_controller()* *(raf-*
con.gui.controllers.utils.extended_controller.ExtendedController *(raf-*
method), 178 *con.gui.controllers.utils.extended_controller.ExtendedController*
relieve_model() *(raf-* *method), 178* *remove_core_element()* *(raf-*
con.gui.controllers.utils.extended_controller.ExtendedController *(raf-*
method), 178 *con.gui.controllers.global_variable_manager.GlobalVariableManager*

method), 182	
remove_core_element()	(raf- con.gui.controllers.state_editor.data_flows.StateDataFlowsListBody) olfer method), 169
remove_core_element()	(raf- con.gui.controllers.state_editor.outcomes.StateOutcomesListCatcher) 143 method), 172
remove_core_element()	(raf- con.gui.controllers.state_editor.scoped_variable_library_remove_variable) Controller method), 167
remove_core_element()	(raf- con.gui.controllers.state_editor.transitions.StateTransitionEditor) rafcon.gui.mygaphas.items.state.StateView method), 175
remove_core_object_from_state()	(raf- con.gui.action.Action static method), 263
remove_data_flow()	(raf- con.core.states.container_state.ContainerState method), 137
remove_data_flows_with_data_port_id()	(raf- con.core.states.container_state.ContainerState method), 138
remove_data_flows_with_data_port_id()	(raf- con.core.states.state.State method), 151
remove_income()	(rafcon.core.states.state.State method), 151
remove_income()	(raf- con.gui.mygaphas.items.state.StateView method), 250
remove_input_data_port()	(raf- con.core.states.library_state.LibraryState method), 143
remove_input_data_port()	(raf- con.core.states.state.State method), 151
remove_input_port()	(raf- con.gui.mygaphas.items.state.StateView method), 250
remove_keep_rect_within_constraint_from_parent()	(rafcon.gui.mygaphas.items.state.StateView method), 250
remove_logging_view()	(raf- con.utils.log_helpers.LoggingViewHandler class method), 277
remove_obsolete_folders()	(in module raf- con.core.storage.storage), 155
remove_outcome()	(raf- con.core.states.library_state.LibraryState method), 143
remove_outcome()	(rafcon.core.states.state.State method), 151
remove_outcome()	(raf- con.gui.mygaphas.items.state.StateView method), 250
remove_outcome_hook()	(raf- con.core.states.container_state.ContainerState
	method), 138
	(raf- remove_outcome_hook() (rafcon.core.states.state.State
	con.gui.controllers.state_editor.data_flows.StateDataFlowsListBody) olfer
	remove_output_data_port() (raf-
	con.core.states.library_state.LibraryState
	remove_output_data_port() (raf-
	con.core.states.state.State method), 151
	remove_output_data_port() (raf-
	con.gui.mygaphas.items.state.StateView
	remove_output_data_port() (raf-
	method), 250
	remove_rafcon_instance_lock_file() (in module rafcon.gui.models.auto_backup), 222
	remove_scoped_variable()
	(raf- con.core.states.container_state.ContainerState method), 138
	remove_scoped_variable()
	(raf- con.gui.mygaphas.items.state.StateView method), 250
	remove_scoped_variables_tab()
	(raf- con.gui.views.state_editor.state_editor.StateEditorView method), 224
	remove_semantic_data()
	(raf- con.core.states.state.State method), 151
	remove_shortcuts()
	(raf- con.gui.shortcut_manager.ShortcutManager method), 270
	remove_source_tab()
	(raf- con.gui.views.state_editor.state_editor.StateEditorView method), 224
	remove_specific_model()
	(raf- con.gui.models.state.StateModel method), 204
	remove_state()
	(raf- con.core.states.barrier_concurrency_state.BarrierConcurrencyState method), 129
	remove_state()
	(raf- con.core.states.container_state.ContainerState method), 138
	remove_transition()
	(raf- con.core.states.container_state.ContainerState method), 138
	remove_tree_children()
	(raf- con.gui.controllers.state_machine_tree.StateMachineTreeControl method), 191
	RemoveObjectAction (class in rafcon.gui.action), 264
	rename() (rafcon.gui.controllers.state_editor.overview.StateOverviewCont method), 173
	rename() (rafcon.gui.controllers.state_editor.state_editor.StateEditorCont method), 170
	rename_library()
	(in module raf- con.gui.helpers.state_machine), 241

rename_library_root() (in module `raf-con.gui.helpers.state_machine`), 241
 rename_selected_state() (raf-
`con.gui.controllers.states_editor.StatesEditorController` method), 194
 replace_all_libraries_by_template() (in module `rafcon.gui.helpers.state_machine`), 241
 reset() (`rafcon.gui.models.modification_history.ModificationsHistory` method), 229
 reset() (`rafcon.utils.timer.Timer` method), 279
 reset_clipboard_mapping_dicts() (raf-
`con.gui.clipboard.Clipboard` method), 269
 reset_from_port() (raf-
`con.gui.mygaphas.items.line.PerpLine` method), 245
 reset_port_for_handle() (raf-
`con.gui.mygaphas.items.connection.ConnectionView` method), 245
 reset_to_history_id() (raf-
`con.gui.models.modification_history.ModificationsHistory` method), 221
 reset_to_port() (raf-
`con.gui.mygaphas.items.line.PerpLine` method), 245
 resize_all_children() (raf-
`con.gui.mygaphas.items.state.StateView` method), 250
 resolve_constraint() (raf-
`con.gui.mygaphas.canvas.MyCanvas` method), 257
 resolve_item_constraints() (raf-
`con.gui.mygaphas.canvas.MyCanvas` method), 257
 resources
 module, 278
 restore_cursor_position() (raf-
`con.gui.views.logging_console.LoggingConsoleView` method), 229
 ReturnItem (class in `raf-con.core.execution.execution_history_items`), 116
 RIGHT (`rafcon.gui.mygaphas.utils.enums.SnappedSide` attribute), 251
 root_state (`rafcon.core.state_machine.StateMachine` property), 163
 root_state (`rafcon.gui.models.state_machine.StateMachineModel` property), 218
 root_state_assign() (raf-
`con.gui.models.state_machine.StateMachineModel` method), 218
 root_state_changed() (raf-
`con.gui.controllers.states_editor.StatesEditorController` method), 194
 root_state_model_after_change() (raf-
`con.gui.models.state_machine.StateMachineModel` method), 218
 root_state_model_before_change() (raf-
`con.gui.models.state_machine.StateMachineModel` method), 218
 rotate_and_detach_tab_labels() (raf-
`con.gui.views.main_window.MainWindowView` method), 229
 run() (`rafcon.core.states.barrier_concurrency_state.BarrierConcurrencyState` method), 130
 run() (`rafcon.core.states.concurrency_state.ConcurrencyState` method), 132
 run() (`rafcon.core.states.container_state.ContainerState` method), 138
 run() (`rafcon.core.states.execution_state.ExecutionState` method), 140
 run() (`rafcon.core.states.hierarchy_state.HierarchyState` method), 141
 run() (`rafcon.core.states.library_state.LibraryState` method), 143
 run() (`rafcon.core.states.preemptive_concurrency_state.PreemptiveConcurrencyState` method), 144
 run() (`rafcon.core.states.state.State` method), 151
 run_decider_state() (raf-
`con.core.states.barrier_concurrency_state.BarrierConcurrencyState` method), 130
 run_hook() (in module `rafcon.utils.plugins`), 278
 run_id (`rafcon.core.states.state.State` property), 152
 run_id_generator() (in module `raf-con.core.id_generator`), 159
 run_on_state_machine_execution_finished() (in module `rafcon.utils.plugins`), 278
 run_post_inits() (in module `rafcon.utils.plugins`), 278
 run_pre_inits() (in module `rafcon.utils.plugins`), 278

S

save_all_libraries() (in module `raf-con.gui.helpers.state_machine`), 241
 save_file_data() (raf-
`con.gui.controllers.state_editor.source_editor.SourceEditorController` method), 167
 save_folder_cmd_line() (in module `raf-con.core.interface`), 160
 save_library() (in module `raf-con.gui.helpers.state_machine`), 241
 save_library_config() (in module `raf-con.gui.helpers.state_machine`), 241
 save_library_dependencies() (in module `raf-con.gui.helpers.state_machine`), 241
 save_open_libraries() (in module `raf-con.gui.helpers.state_machine`), 241
 save_script_file_for_state_and_source_path() (in module `rafcon.core.storage.storage`), 155

save_selected_state_as() (in module raf-
con.gui.helpers.state_machine), 242
save_semantic_data_for_state() (in module raf-
con.core.storage.storage), 155
save_state_machine() (in module raf-
con.gui.helpers.state_machine), 242
save_state_machine_as() (in module raf-
con.gui.helpers.state_machine), 242
save_state_machine_to_path() (in module raf-
con.core.storage.storage), 156
save_state_recursively() (in module raf-
con.core.storage.storage), 156
scoped_data (rafcon.core.states.container_state.ContainerState
property), 138
scoped_variable
 module, 124
scoped_variable
 (raf-
 con.gui.models.scoped_variable.ScopedVariableModel
 property), 211
scoped_variable()
 (raf-
 con.gui.mygaphas.items.state.StateView
 method), 250
scoped_variable_list
 module, 165
scoped_variables
 (raf-
 con.core.states.container_state.ContainerState
 property), 138
scoped_variables
 (raf-
 con.gui.models.container_state.ContainerModelState
 property), 206
scoped_variables
 (raf-
 con.gui.models.selection.Selection
 property), 214
scoped_variables
 (raf-
 con.gui.mygaphas.items.state.StateView
 property), 250
scoped_variables_changed()
 (raf-
 con.gui.controllers.state_editor.scoped_variable_list.ScopedVariableListController
 method), 167
ScopedData (class in rafcon.core.state_elements.scope),
 124
ScopedDataItem (class in raf-
 con.core.execution.execution_history_items),
 116
ScopedVariable (class in raf-
 con.core.state_elements.scope), 125
ScopedVariableAction (class in rafcon.gui.action),
 265
ScopedVariableListController (class in raf-
 con.gui.controllers.state_editor.scoped_variable_list), 165
ScopedVariableModel (class in raf-
 con.gui.models.scoped_variable), 211
ScopedVariablePortView (class in raf-
 con.gui.mygaphas.items.ports), 248
ScopesVariablesListView (class in raf-
 con.gui.views.state_editor.scoped_variables_list),
 225
script
 module, 161
Script (class in rafcon.core.script), 161
script (rafcon.core.script.Script property), 161
script (rafcon.core.states.execution_state.ExecutionState
property), 140
script_text (rafcon.core.states.execution_state.ExecutionState
property), 140
script_text (rafcon.gui.action.StateImage attribute),
 266
script_text_changed()
 (raf-
 con.gui.controllers.states_editor.StatesEditorController
 method), 194
scroll_to_bottom()
 (raf-
 con.gui.controllers.state_editor.description_editor.DescriptionEditor
 method), 170
scroll_to_cursor_onscreen()
 (raf-
 con.gui.views.logging_console.LoggingConsoleView
 method), 229
scroll_to_cursor_onscreen()
 (raf-
 con.gui.views.utils.editor.EditorView method),
 227
search_in_share_folders() (in module raf-
 con.utils.resources), 278
select_item()
 (rafcon.gui.mygaphas.view.ExtendedGtkView
 method), 262
select_library_tree_element_of_lib_tree_path()
 (rafcon.gui.controllers.library_tree.LibraryTreeController
 method), 185
select_library_tree_element_of_library_state_model()
 (rafcon.gui.controllers.library_tree.LibraryTreeController
 method), 185
selected (rafcon.gui.mygaphas.items.state.StateView
property), 262
selected_state_toggle_is_start_state()
 (in
 module rafcon.gui.helpers.state_machine), 242
Selection (class in rafcon.gui.models.selection), 211
selection (rafcon.gui.models.state_machine.StateMachineModel
attribute), 218
selection_changed()
 (raf-
 con.gui.controllers.state_machine_tree.StateMachineTreeController
 method), 191
selection_changed_signal
 (raf-
 con.gui.models.selection.Selection
 property), 214
selection_notification()
 (raf-
 con.gui.controllers.states_editor.StatesEditorController

<code>method), 194</code>	
<code>semantic_data (rafcon.core.states.state.State property), 152</code>	
<code>semantic_data (rafcon.gui.action.StateImage attribute), 266</code>	
<code>separate_folder_path_and_file_name() (in module rafcon.utils.filesystem), 273</code>	
<code>set() (rafcon.gui.models.selection.Selection method), 214</code>	
<code>set_after() (rafcon.gui.action.AbstractAction method), 262</code>	
<code>set_after() (rafcon.gui.action.AddObjectAction method), 263</code>	
<code>set_after() (rafcon.gui.action.RemoveObjectAction method), 265</code>	
<code>set_after() (rafcon.gui.action.StateAction method), 265</code>	
<code>set_after() (rafcon.gui.action.StateElementAction method), 266</code>	
<code>set_button_children_size_request() (in module rafcon.gui.helpers.label), 233</code>	
<code>set_config_value() (rafcon.core.config.ObservableConfig method), 157</code>	
<code>set_cursor_on_line_with_string() (rafcon.gui.views.logging_console.LoggingConsoleView method), 229</code>	
<code>set_cursor_position() (rafcon.gui.views.logging_console.LoggingConsoleView method), 229</code>	
<code>set_cursor_position() (rafcon.gui.views.utils.editor.EditorView method), 227</code>	
<code>set_default_paned_positions() (rafcon.gui.views.state_editor.state_editor.StateEditorView method), 224</code>	
<code>set_dict() (rafcon.utils.vividict.Vividict method), 280</code>	
<code>set_editor_lock() (rafcon.gui.controllers.state_editor.source_editor.SourceEditorView method), 167</code>	
<code>set_enable_flag_keep_rect_within_constraints() (rafcon.gui.mygaphas.items.state.StateView method), 250</code>	
<code>set_enabled() (rafcon.gui.views.utils.editor.EditorView method), 227</code>	
<code>set_enables() (rafcon.gui.views.logging_console.LoggingConsoleView method), 229</code>	
<code>set_icon_and_text_box_of_menu_item() (in module rafcon.gui.helpers.label), 233</code>	
<code>set_input_runtime_value() (rafcon.core.states.library_state.LibraryState method), 144</code>	
<code>set_label_markup() (in module rafcon.gui.helpers.label), 233</code>	
<code>set_menu_item_accelerator() (rafcon.gui.views.menu_bar.MenuBarView method), 230</code>	
<code>set_menu_item_icon() (rafcon.gui.views.menu_bar.MenuBarView method), 230</code>	
<code>set_menu_item_sensitive() (rafcon.gui.views.menu_bar.MenuBarView method), 230</code>	
<code>set_nearest_border() (rafcon.gui.mygaphas.constraint.PortRectConstraint method), 259</code>	
<code>set_notebook_title() (in module rafcon.gui.helpers.label), 233</code>	
<code>set_observable_action_signal() (rafcon.gui.models.abstract_state.AbstractStateModel method), 201</code>	
<code>set_observable_action_signal() (rafcon.gui.models.state_machine.StateMachineModel method), 218</code>	
<code>set_observable_change_count() (rafcon.gui.models.modification_history.ModificationsHistoryModel method), 221</code>	
<code>set_observable_data_flow() (rafcon.gui.models.data_flow.DataFlowModel method), 210</code>	
<code>set_observable_data_flows() (rafcon.gui.models.container_state.ContainerStateModel method), 206</code>	
<code>set_observable_data_port() (rafcon.gui.models.data_port.DataPortModel method), 211</code>	
<code>set_observable_destruction_signal() (rafcon.gui.models.abstract_state.AbstractStateModel method), 201</code>	
<code>set_observable_destruction_signal() (rafcon.gui.models.state_element.StateElementModel method), 208</code>	
<code>set_observable_destruction_signal() (rafcon.gui.models.state_machine.StateMachineModel method), 218</code>	
<code>set_observable_focus_signal() (rafcon.gui.models.selection.Selection method), 214</code>	
<code>set_observable_income() (rafcon.gui.models.abstract_state.AbstractStateModel method), 201</code>	
<code>set_observable_income() (rafcon.gui.models.logical_port.IncomeModel method), 215</code>	
<code>set_observable_input_data_ports() (rafcon.gui.models.abstract_state.AbstractStateModel method), 201</code>	
<code>set_observable_is_start() (rafcon.gui.models.state_machine.StateMachineModel method), 218</code>	

`con.gui.models.abstract_state.AbstractStateModel
method), 201`

`set_observable_meta_signal() (raf-
con.gui.models.abstract_state.AbstractStateModel
method), 201`

`set_observable_meta_signal() (raf-
con.gui.models.state_element.StateElementModel
method), 208`

`set_observable_meta_signal() (raf-
con.gui.models.state_machine.StateMachineModel
method), 218`

`set_observable_modifications() (raf-
con.gui.models.modification_history.ModificationsHistory
method), 221`

`set_observable_ongoing_complex_actions() (raf-
con.gui.models.state_machine.StateMachineModel
method), 218`

`set_observable_outcome() (raf-
con.gui.models.logical_port.OutcomeModel
method), 216`

`set_observable_outcomes() (raf-
con.gui.models.abstract_state.AbstractStateModel
method), 201`

`set_observable_output_data_ports() (raf-
con.gui.models.abstract_state.AbstractStateModel
method), 201`

`set_observable_root_state() (raf-
con.gui.models.state_machine.StateMachineModel
method), 218`

`set_observable_scoped_variable() (raf-
con.gui.models.scoped_variable.ScopedVariableModel
method), 211`

`set_observable_scoped_variables() (raf-
con.gui.models.container_state.ContainerStateModel
method), 206`

`set_observable_selection_changed_signal()
(rafcon.gui.models.selection.Selection
method), 214`

`set_observable_sm_selection_changed_signal()
(rafcon.gui.models.state_machine.StateMachineModel
method), 218`

`set_observable_state() (raf-
con.gui.models.abstract_state.AbstractStateModel
method), 201`

`set_observable_state_action_signal() (raf-
con.gui.models.state_machine.StateMachineModel
method), 218`

`set_observable_state_machine() (raf-
con.gui.models.state_machine.StateMachineModel
method), 218`

`set_observable_state_meta_signal() (raf-
con.gui.models.state_machine.StateMachineModel
method), 218`

`set_observable_states() (raf-`

`con.gui.models.container_state.ContainerStateModel
method), 206`

`set_observable_transition() (raf-
con.gui.models.transition.TransitionModel
method), 209`

`set_observable_transitions() (raf-
con.gui.models.container_state.ContainerStateModel
method), 206`

`set_output_runtime_value() (raf-
con.core.states.library_state.LibraryState
method), 144`

`set_port_for_handle() (raf-
con.gui.mygaphas.items.connection.ConnectionView
method), 245`

`set_script_text() (raf-
con.gui.controllers.utils.editor.EditorController
method), 176`

`set_script_without_compilation() (raf-
con.core.script.Script
method), 161`

`set_semantic_dictionary_list() (raf-
con.gui.clipboard.Clipboard
method), 269`

`set_start_state() (raf-
con.core.states.container_state.ContainerState
method), 138`

`set_tab_label_texts() (in module raf-
con.gui.controllers.states_editor), 195`

`set_text() (rafcon.gui.views.utils.editor.EditorView
method), 227`

`set_timed_thread() (raf-`

`set_use_input_runtime_value() (raf-
con.gui.models.auto_backup.AutoBackupModel
method), 222`

`set_use_output_runtime_value() (raf-
con.core.states.library_state.LibraryState
method), 144`

`set_window_size_and_position() (in module raf-
con.gui.helpers.label), 233`

`setup_backward_run() (rafcon.core.states.state.State
method), 152`

`setup_canvas() (raf-
con.gui.mygaphas.items.state.StateView
method), 250`

`setup_forward_or_backward_execution() (raf-
con.core.states.concurrency_state.ConcurrencyState
method), 132`

`setup_i18n() (in module rafcon.utils.i18n), 275`

`setup_run() (rafcon.core.states.state.State
method), 138`

`ShortcutManager (class in raf-
con.gui.shortcut_manager), 269`

show_aborted_preempted() (raf-
 con.gui.controllers.menu_bar.MenuBarController singular_form() (in module rafcon.gui.clipboard), 269
 method), 188
 show_connection (raf-
 con.gui.mygaphas.items.connection.DataFlowView property), 245
 show_connection (raf-
 con.gui.mygaphas.items.connection.TransitionView solve_for() (rafcon.gui.mygaphas.constraint.BorderWidthConstraint property), 246
 show_content() (raf-
 con.gui.controllers.state_machine_tree.StateMachineTreeController), 258
 method), 191
 show_content() (raf-
 con.gui.models.library_state.LibraryStateModel solve_for() (rafcon.gui.mygaphas.constraint.KeepPointWithinConstraint method), 258
 method), 207
 show_content() (raf-
 con.gui.mygaphas.items.state.StateView solve_for() (rafcon.gui.mygaphas.constraint.KeepRelativePositionConstraint method), 251
 method), 259
 show_content_changed() (raf-
 con.gui.controllers.state_editor.overview.StateOverviewController
 method), 173
 show_content_changed() (raf-
 con.gui.controllers.state_editor.state_editor.StateEditorController property), 170
 method), 170
 show_data_flows_toggled_shortcut() (raf-
 con.gui.controllers.menu_bar.MenuBarController source_text (rafcon.gui.controllers.utils.editor.EditorController property), 167
 method), 188
 show_data_port_label (raf-
 con.gui.mygaphas.items.state.StateView SourceEditorController (class in raf-
 property), 251
 con.gui.controllers.state_editor.source_editor), 167
 show_data_values_toggled_shortcut() (raf-
 con.gui.controllers.menu_bar.MenuBarController SourceEditorView (class in raf-
 method), 188
 con.gui.views.state_editor.source_editor), 225
 show_notice() (in module rafcon.core.interface), 160
 show_notice_func() (in module rafcon.core.interface), 160
 show_transitions_toggled_shortcut() (raf-
 con.gui.controllers.menu_bar.MenuBarController TransitionSegment
 method), 188
 shutdown() (rafcon.core.execution.base_execution_history.BaseExecutionHistory start() (in module rafcon.utils.profiler), 278
 method), 114
 shutdown() (rafcon.core.execution.in_memory_execution_history.startOnMemoryExecutionHistory
 method), 118
 side (rafcon.gui.mygaphas.items.ports.PortView start() (rafcon.core.states.state.State
 property), 247
 method), 152
 single_widget_window (raf-
 module, 178
 con.core.states.concurrency_stateConcurrencyState
 singleton
 method), 162
 SingleWidgetWindowController (class in raf-
 con.gui.controllers.utils.single_widget_window), start_child_states() (raf-
 178
 con.core.states.concurrency_state.ConcurrencyState
 Singleton
 method), 132
 SingleWidgetWindowView (class in raf-
 con.gui.views.utils.single_widget_window), start_move() (rafcon.gui.mygaphas.guide.GuidedStateInMotion
 method), 259
 con.core.states.state.State
 method), 221
 SingleWidgetWindowView (class in raf-
 con.gui.views.utils.single_widget_window), start_new_action() (raf-
 method), 221
 con.gui.models.modification_history.ModificationsHistoryModel
 start_new_action_old() (raf-

con.gui.models.modification_history.ModificationsHistory
method), 221

start_state_id (raf-
con.core.states.container_state.ContainerState
property), 138

started (rafcon.core.states.state.State property), 152

started_in_virtualenv() (in module raf-
con.utils.installation), 275

started_without_installation() (in module raf-
con.utils.installation), 275

state
module, 145

State (class in rafcon.core.states.state), 145

state (rafcon.gui.models.abstract_state.AbstractStateModel
property), 201

state_action_signal (raf-
con.gui.models.state_machine.StateMachineModel
property), 218

state_action_signal() (raf-
con.gui.controllers.state_machine_tree.StateMachineTreeController
method), 191

state_action_signal() (raf-
con.gui.controllers.states_editor.StatesEditorController
method), 194

state_action_signal() (raf-
con.gui.controllers.state_machine_tree.StateMachineTreeController
method), 191

state_action_signal() (raf-
con.gui.controllers.states_editor.StatesEditorController
method), 194

state_action_signal() (raf-
con.gui.models.modification_history.ModificationsHistory
method), 221

state_action_signal() (raf-
con.gui.models.state_machine.ComplexActionObserver
method), 216

state_copy (rafcon.core.states.library_state.LibraryState
property), 144

state_copy (rafcon.gui.models.library_state.LibraryState
attribute), 207

state_counter (rafcon.gui.models.abstract_state.AbstractStateModel
attribute), 201

state_data_flows
module, 168

state_destruction() (raf-
con.gui.controllers.state_editor.state_editor.StateEditorController
method), 170

state_editor
module, 170

state_element
module, 127

state_element_attrs (rafcon.core.states.state.State
attribute), 152

state_element_id (raf-
con.core.state_elements.data_flow.DataFlow
property), 121

state_element_id (raf-
con.core.state_elements.data_port.DataPort
property), 122

state_element_id (raf-

makecore.state_elements.logical_port.Income
property), 123

state_element_id (raf-
con.core.state_elements.logical_port.Outcome
property), 124

state_element_id (raf-
con.core.state_elements.scope.ScopedData
property), 125

state_element_id (raf-
con.core.state_elements.state_element.StateElement
property), 128

state_element_id (raf-
con.core.state_elements.transition.Transition
property), 127

state_element_to_dict() (raf-
con.core.state_elements.data_flow.DataFlow
static method), 121

state_element_to_dict() (raf-
con.core.state_elements.data_port.DataPort
static method), 122

state_element_to_dict() (raf-
con.core.state_elements.logical_port.Income
static method), 123

state_element_to_dict() (raf-
con.core.state_elements.logical_port.Outcome
static method), 124

state_element_to_dict() (raf-
con.core.state_elements.scope.ScopedData
static method), 125

state_element_to_dict() (raf-
con.core.state_elements.state_element.StateElement
static method), 126

state_element_to_dict() (raf-
con.core.state_elements.transition.Transition
static method), 127

state_execution_status (raf-
con.core.states.state.State property), 152

state_icons
module, 196

state_id (rafcon.core.states.state.State property), 152

state_id_generator() (in module raf-
con.core.id_generator), 159

state_machine
module, 162

state_machine (rafcon.gui.models.state_machine.StateMachineModel
property), 218

state_machine_id (raf-
con.core.state_machine.StateMachine
attribute), 163

state_machine_manager
module, 164

state_machine_manager_notification() (raf-
con.gui.controllers.modification_history.ModificationsHistory.*Modification*) static method), 144
state_machine_manager_notification() (raf-
con.gui.controllers.state_machine_tree.StateMachineTreeController) static method), 152
state_machine_manager_notification() (raf-
con.gui.controllers.state_machine_tree.StateMachineTreeController) state_transitions
method), 191
StateAction (class in *rafcon.gui.action*), 265
state_machine_manager_notification() (raf-
con.gui.controllers.states_editor.StatesEditorController (class in *rafcon.gui.controllers.states_editor*),
method), 194
state_machine_model (raf-
con.gui.models.modification_history.ModificationsHistory.*Modification*) static attribute), 221
StateDataFlowsListController (class in *rafcon.gui.controllers.state_editor.data_flows*),
state_machine_model_after_change() (raf-
con.gui.models.state_machine.StateMachineModel) 168
StateDataFlowsListView (class in *rafcon.gui.controllers.state_editor.data_flows*),
method), 218
state_machine_notification() (raf-
con.gui.controllers.state_machine_tree.StateMachineTreeController) static attribute), 225
StateEditorController (class in *rafcon.gui.controllers.state_editor.state_editor*),
state_machine_status
 module, 117
state_machine_to_dict() (raf-
con.core.state_machine.StateMachine) static method), 163
state_machine_tree
 module, 190
state_machines_del_notification() (raf-
con.gui.controllers.states_editor.StatesEditorController) static method), 194
StateExecutionStatus (in module *rafcon.core.states.state*), 153
state_machines_set_notification() (raf-
con.gui.controllers.states_editor.StatesEditorController) static method), 194
StateIconController (class in *rafcon.gui.controllers.state_icons*), 196
StateImage (class in *rafcon.gui.action*), 266
state_meta_signal
con.gui.models.state_machine.StateMachineMode property), 218
StateMachine (class in *rafcon.core.state_machine*), 162
StateMachineAction (class in *rafcon.gui.action*), 266
StateMachineExecutionStatus (in module *rafcon.core.execution.execution_status*), 117
state_meta_update() (raf-
con.gui.controllers.state_machine_tree.StateMachineTreeController) static method), 191
state_outcomes
 module, 171
state_overview
 module, 172
state_path (*rafcon.gui.action.StateImage* attribute), 266
STATE_PATH_STORAGE_ID (raf-
con.gui.controllers.state_machine_tree.StateMachineTreeController) static attribute), 190
state_reference
con.core.execution.execution_history_items.HistoryItem property), 116
StateMachineTreeController (class in *rafcon.gui.controllers.state_machine_tree*), 190
StateMachineTreeView (class in *rafcon.gui.views.state_machine_tree*), 230
StateModel (class in *rafcon.gui.models.state*), 202
StateOutcomesEditorController (class in *rafcon.gui.controllers.state_editor.outcomes*), 171
StateOutcomesEditorView (class in *rafcon.gui.views.state_editor.outcomes*), 226
StateOutcomesListController (class in *rafcon.gui.controllers.state_editor.outcomes*), 171

con.gui.controllers.state_editor.outcomes), 171
StateOutcomesTreeView (class in raf-con.gui.views.state_editor.outcomes), 226
StateOverviewController (class in raf-con.gui.controllers.state_editor.overview), 172
StateOverviewView (class in raf-con.gui.views.state_editor.overview), 226
states (rafcon.core.states.barrier_concurrency_state.BarrierConcurrencyState), 185
 property), 130
states (rafcon.core.states.container_state.ContainerState
 property), 139
states (rafcon.gui.models.container_state.ContainerStateModel
 property), 206
states (rafcon.gui.models.selection.Selection property), 214
states_editor
 module, 192
states_update() (raf-con.gui.controllers.state_machine_tree.StateMachineTreeController
 method), 191
states_update_before() (raf-con.gui.controllers.state_machine_tree.StateMachineTreeController
 method), 191
StatesEditorController (class in raf-con.gui.controllers.states_editor), 192
StatesEditorView (class in raf-con.gui.views.states_editor), 231
StateTransitionsEditorController (class in raf-con.gui.controllers.state_editor.transitions), 173
StateTransitionsEditorView (class in raf-con.gui.views.state_editor.transitions), 226
StateTransitionsListController (class in raf-con.gui.controllers.state_editor.transitions), 173
StateTransitionsListView (class in raf-con.gui.views.state_editor.transitions), 226
StateType (in module rafcon.core.states.state), 153
StateView (class in rafcon.gui.mygaphas.items.state), 249
stop() (in module rafcon.utils.profiler), 278
stop() (rafcon.core.execution.consumers.abstract_execution.ConsumerAbstractExecutionHistoryConsumer
 method), 119
stop() (rafcon.utils.timer.Timer method), 279
stop_consumers() (raf-con.core.execution.consumer_manager.ExecutionHistoryConsumerManager
 method), 114
stop_move() (rafcon.gui.mygaphas.guide.GuidedStateInMotion
 method), 259
stop_worker_thread() (raf-con.core.execution.consumer_manager.ExecutionHistoryConsumerManager
 method), 114
storage
 module, 153
storage_utils
 module, 279
store_cursor_position() (raf-con.gui.views.logging_console.LoggingConsoleView
 method), 229
store_expansion_state() (raf-con.gui.controllers.library_tree.LibraryTreeController
 method), 185
store_expansion_state() (raf-con.gui.controllers.state_machine_tree.StateMachineTreeController
 method), 191
store_meta_data() (raf-con.gui.models.abstract_state.AbstractStateManager
 method), 201
store_meta_data() (raf-con.gui.models.container_state.ContainerStateModel
 method), 206
store_meta_data() (raf-con.gui.controllers.state_machine_tree.StateMachineTreeController
 method), 218
store_related_elements() (raf-con.gui.controllers.state_machine_tree.StateMachineTreeController
 method), 265
sub_menus (rafcon.gui.views.menu_bar.MenuBarView
 attribute), 230
substitute_as_library_clicked() (raf-con.gui.controllers.library_tree.LibraryTreeController
 method), 185
substitute_as_template_clicked() (raf-con.gui.controllers.library_tree.LibraryTreeController
 method), 185
substitute_dict (rafcon.gui.action.StateAction
 attribute), 265
substitute_selected_library_state_with_template()
 (in module rafcon.gui.helpers.state_machine), 243
substitute_selected_state() (in module raf-con.gui.helpers.state_machine), 243
substitute_selected_state_and_use_choice_dialog()
 (in module rafcon.gui.helpers.state_machine), 243
substitute_state() (raf-con.core.states.container_state.ContainerState
 method), 185
substitute_state() (raf-con.core.states.state.State
 method), 236
substitute_state() (raf-con.core.states.state.State
 method), 243
substitute_state_as() (in module raf-con.core.states.state.State
 method), 206
supports_saving_state_names (raf-con.core.states.state.State
 method), 236

con.core.state_machine.StateMachine *prop-* *TO_OUTCOME_STORAGE_ID* (*raf-*
erty), 163 *con.gui.controllers.state_editor.transitions.StateTransitionsListCo*
suppress_new_root_state_model_one_time (*raf-*
con.gui.models.state_machine.StateMachineMode
to_port (*rafcon.gui.mygaphas.items.line.PerpLine prop-*
attribute), 219 *erty*), 245
synchronized_redo() (*raf-*
con.gui.models.modification_history.ModificationsHistoryMod
property), 121
synchronized_undo() (*raf-*
con.gui.models.modification_history.ModificationT
method), 221 *TO_STATE_STORAGE_ID* (*raf-*
attribute), 168
T
Timer (*class in rafcon.utils.timer*), 279
timestamp (*rafcon.core.state_elements.scope.ScopedData*
property), 125
tmp_connect() (*rafcon.gui.mygaphas.items.ports.PortView*
method), 247
tmp_disconnect() (*raf-*
con.gui.mygaphas.items.ports.PortView
method), 248
tmp_file (*rafcon.gui.controllers.state_editor.SourceEditor*
attribute), 167
tmp_meta_storage (*raf-*
con.gui.models.modification_history.ModificationsHistoryModel
property), 221 *TOOL_TIP_STORAGE_ID* (*raf-*
module 195
to_dict() (*rafcon.core.execution.execution_history_items.CallItem*
method), 115 *attribute*), 179
to_dict() (*rafcon.core.execution.execution_history_items.CallItem*
method), 115 *TOOL_TIP_STORAGE_ID* (*raf-*
con.gui.controllers.library_tree.LibraryTreeController
module 195
to_dict() (*rafcon.core.execution.execution_history_items.HistoryItem*
method), 116 *TOOL_TIP_TEXT* (*rafcon.gui.controllers.execution_history.ExecutionHist*
attribute), 184
to_dict() (*rafcon.core.execution.execution_history_items.ReturnItem*
method), 116 *attribute*), 179
to_dict() (*rafcon.core.execution.execution_history_items.ScopedData*
method), 116 *ToolBarController* (*class* *in* *raf-*
module 195
to_dict() (*rafcon.core.execution.execution_history_items.StateElement*
method), 128 *ToolBarView* (*class in rafcon.gui.views.tool_bar*), 231
to_dict() (*rafcon.core.state_machine.StateMachine*
method), 163 *TOP_MarginStorage* (*rafcon.gui.mygaphas.utils.enums.SnappedSide* *at-*
attribute), 251
to_dict() (*rafcon.core.states.state.State* *method*), 152
to_dict() (*rafcon.utils.vividict.Vividict* *method*), 280
to_handle() (*rafcon.gui.mygaphas.items.line.PerpLine*
method), 245
to_key (*rafcon.core.state_elements.data_flow.DataFlow*
property), 121
TO_KEY_STORAGE_ID (*raf-*
con.gui.controllers.state_editor.data_flows.StateDataFlowsListCo
attribute), 168
to_outcome (*rafcon.core.state_elements.transition.Transition*
property), 127 *transition* (*class* *in* *raf-*
con.core.state_elements.transition), 126
transition_id (*rafcon.core.state_elements.transition.Transition*
property), 127 *transition* (*rafcon.gui.models.transition.TransitionModel*
attribute), 209
TransitionAction (*class in rafcon.gui.action*), 267
TransitionModel (*class* *in* *raf-*

con.gui.models.transition), 209
TransitionPlaceholderView (class in raf- undocked_window
con.gui.mygaphas.items.connection), 246 module, 197
transitions (rafcon.core.states.container_state.ContainerState UndockedWindowController (class in raf-
property), 139 ungroup_selected_state() (in module raf-
transitions (rafcon.gui.models.container_state.ContainerStateModel
property), 206 ungroup_state() (in module rafcon.gui.helpers.state),
transitions (rafcon.gui.models.selection.Selection 236
property), 214 ungroup_state() (raf-
TransitionSegment (class in raf- con.core.states.container_state.ContainerState
con.gui.mygaphas.segment), 260 method), 139
TransitionView (class in raf- ungroup_state() (raf-
con.gui.mygaphas.items.connection), 246 con.gui.models.container_state.ContainerStateModel
transparency (rafcon.gui.mygaphas.items.state.NameView method), 206
property), 249 unregister() (rafcon.core.execution.consumers.abstract_execution_histo-
transparency (rafcon.gui.mygaphas.items.state.StateView ry method), 119
property), 251 unregister() (rafcon.core.execution.consumers.file_system_consumer.File
treeView (rafcon.gui.views.state_editor.linkage_overview.LinkageOverViewLogicView
property), 224 unregister_actions() (raf-
trigger_action() (raf- con.gui.controllers.utils.extended_controller.ExtendedController
con.gui.shortcut_manager.ShortcutManager method), 178
method), 270 unregister_consumer() (raf-
type_helpers con.core.execution.consumer_manager.ExecutionHistoryConsum-
module, 279 er
type_inherits_of_type() (in module raf- unregister_view() (raf-
con.utils.type_helpers), 279 imSelectable) (raf-
TYPE_NAME_STORAGE_ID (raf- con.gui.mygaphas.view.ExtendedGtkView
attribute), 190 method), 262
type_related_list_name_dict (raf- unselect_item() (raf-
con.gui.action.CoreObjectIdentifier attribute), con.gui.mygaphas.view.ExtendedGtkView
263 method), 262
update() (rafcon.gui.controllers.execution_history.ExecutionHistoryTreeC
undo() (rafcon.gui.action.AbstractAction method), 262 update() (rafcon.gui.controllers.execution_history.ExecutionHistoryTreeC
undo() (rafcon.gui.action.Action method), 263 update() (rafcon.gui.controllers.library_tree.LibraryTreeController
undo() (rafcon.gui.action.AddObjectAction method), 263 update() (rafcon.gui.controllers.modification_history.ModificationHistory
attribute), 190 update() (rafcon.gui.controllers.state_editor.data_flows.StateDataFlowsL
undo() (rafcon.gui.action.DataFlowAction method), 264 update() (rafcon.gui.controllers.state_editor.outcomes.StateOutcomesList
undo() (rafcon.gui.action.DataPortAction method), 264 update() (rafcon.gui.controllers.state_editor.transitions.StateTransitionsL
undo() (rafcon.gui.action.MetaDataAction method), 264 update() (rafcon.gui.controllers.state_machine_tree.StateMachineTreeCo
undo() (rafcon.gui.action.OutcomeAction method), 264
undo() (rafcon.gui.action.RemoveObjectAction method), 265 update_auto_scroll_mode() (raf-
attribute), 263 update() (rafcon.gui.controllers.modification_history.ModificationHistoryT
undo() (rafcon.gui.action.StateAction method), 266 update() (rafcon.gui.controllers.state_machine_tree.StateMachineTreeCo
undo() (rafcon.gui.action.StateMachineAction method), 266 update() (rafcon.gui.controllers.state_machine_tree.StateMachineTreeCo
attribute), 266 update() (rafcon.gui.controllers.state_machine_tree.StateMachineTreeCo
undo() (rafcon.gui.action.TransitionAction method), 267 update_history_child_is_start() (raf-
attribute), 267 update() (rafcon.gui.controllers.state_machine_tree.StateMachineTreeCo
undo() (rafcon.gui.controllers.modification_history.ModificationHistoryT
attribute), 189 update() (rafcon.gui.controllers.state_machine_tree.StateMachineTreeCo
undo() (rafcon.gui.models.modification_history.ModificationHistoryT
attribute), 220 update() (rafcon.gui.models.state_machine_tree.StateMachineTreeCo
undo() (rafcon.gui.models.modification_history.ModificationsHistoryModel), 206 update() (rafcon.gui.models.state_machine_tree.StateMachineTreeCo
attribute), 221 update_child_models() (raf-

con.gui.models.container_state.ContainerStateModel
 method), 206
update_core_element_lists() (raf-
 con.gui.models.selection.Selection
 214
update_data_flow_from_image() (raf-
 con.gui.action.DataFlowAction
 264
update_data_flows() (in module raf-
 con.gui.controllers.state_editor.data_flows),
 169
update_data_port_from_image() (raf-
 con.gui.action.DataPortAction
 264
update_distance_to_border() (raf-
 con.gui.mygaphas.constraint.PortRectConstraint
 method), 259
update_filtered_buffer() (raf-
 con.gui.controllers.logging_console.LoggingConsoleController
 method), 183
update_font_cache() (in module raf-
 con.utils.installation), 275
update_global_variables_list_store() (raf-
 con.gui.controllers.global_variable_manager.GlobalVariableManager@aphashutils.gap_helper),
 method), 182
update_hash() (rafcon.core.state_elements.state_element.StateElement.gui.mygaphas.utils.gap_helper),
 method), 128
update_hash() (rafcon.core.state_machine.StateMachine
 method), 163
update_hash() (rafcon.core.states.container_state.ContainerState
 method), 139
update_hash() (rafcon.core.states.execution_state.ExecutionState
 method), 140
update_hash() (rafcon.core.states.library_state.LibraryState
 method), 144
update_hash() (rafcon.core.states.state.State
 method), 152
update_hash() (rafcon.gui.models.abstract_state.AbstractStateModel
 method), 201
update_hash() (rafcon.gui.models.container_state.ContainerStateModel
 method), 249
update_hash() (rafcon.gui.models.library_state.LibraryStateModel
 method), 207
update_hash() (rafcon.gui.models.state_element.StateElement
 method), 208
update_hash() (rafcon.gui.models.state_machine.StateMachineModel
 method), 219
update_hash() (rafcon.utils.hashable.Hashable
 method), 274
update_hash_from_dict() (raf-
 con.utils.hashable.Hashable static method),
 275
update_internal_data_base() (raf-
 con.gui.controllers.state_editor.outcomes.StateOutcomesListCont

method), 172
update_internal_tmp_storage() (raf-
 con.gui.models.modification_history.ModificationsHistoryModel
 method), 221
update_is_start() (raf-
 con.gui.models.abstract_state.AbstractStateModel
 method), 201
update_last_backup_meta_data() (raf-
 con.gui.models.auto_backup.AutoBackupModel
 method), 222
update_last_sm_origin_meta_data() (raf-
 con.gui.models.auto_backup.AutoBackupModel
 method), 222
update_list_store() (raf-
 con.gui.controllers.state_editor.outcomes.StateOutcomesListCont
 method), 172
update_maximize_button() (raf-
 con.gui.controllers.top_tool_bar.TopToolBarController
 method), 196
update_meta_data_for_connection_waypoints()
 (in module raf-
 con.gui.mygaphas.utils.gap_helper), 254
update_meta_data_for_name_view() (in module raf-
 con.gui.mygaphas.utils.gap_helper), 255
update_meta_data_for_port() (in module raf-
 con.gui.mygaphas.utils.gap_helper), 255
update_meta_data_for_state_view() (in module raf-
 con.gui.mygaphas.utils.gap_helper), 255
update_meta_data_hash() (raf-
 con.gui.models.abstract_state.AbstractStateModel
 method), 202
update_meta_data_hash() (raf-
 con.gui.models.container_state.ContainerStateModel
 method), 206
update_meta_data_hash() (raf-
 con.gui.models.state_machine.StateMachineModel
 method), 219
update_meta_data_min_size() (raf-
 con.gui.mygaphas.items.state.NameView
 method), 251
update_meta_data_min_size() (raf-
 con.gui.mygaphas.items.state.StateView
 method), 251
update_meta_data_min_size_of_children() (raf-
 con.gui.mygaphas.items.state.StateView
 method), 251
update_models() (rafcon.gui.models.state.StateModel
 method), 204
update_models_recursively() (in module raf-
 con.gui.helpers.state), 236
update_outcome_from_image() (raf-
 con.gui.action.OutcomeAction
 method), 264
update_outsize() (raf-

con.gui.mygaphas.constraint.PortRectConstraint **version** (*rafcon.core.states.library_state.LibraryState*
method), 259 *property*), 144
update_position() (*raf-*
con.gui.mygaphas.constraint.PortRectConstraint
method), 259 *view* (*rafcon.gui.mygaphas.items.line.PerpLine* *prop-*
erty), 245
update_property_from_image() (*raf-*
con.gui.action.StateAction *method*), 266 *view* (*rafcon.gui.mygaphas.items.ports.PortView* *prop-*
erty), 248
update_root_items() (*raf-*
con.gui.mygaphas.canvas.MyCanvas *method*), 257 *view* (*rafcon.gui.mygaphas.items.state.NameView* *prop-*
erty), 249
update_root_state_from_image() (*raf-*
con.gui.action.StateMachineAction *method*), 267 *view* (*rafcon.gui.mygaphas.segment.TransitionSegment*
vividict *property*), 260
update_scoped_variables_with_output_dictionary() *module*, 280
(rafcon.core.states.container_state.ContainerState *vividict* (*class* *in* *rafcon.utils.vividict*)), 280
method), 139 *vividict_to_dict()* (*rafcon.utils.vividict.Vividict*
static method), 280

W

wait_for_interruption() (*raf-*
con.core.states.state.State *method*), 152
wait_for_unpause() (*rafcon.core.states.state.State*
method), 153
wait_for_update() (*raf-*
con.gui.mygaphas.canvas.MyCanvas *method*),
257
waypoints (*rafcon.gui.mygaphas.items.line.PerpLine*
property), 245
width (*rafcon.gui.mygaphas.connector.RectanglePointPort*
property), 257
worker() (*rafcon.core.execution.consumers.abstract_execution_history_cc*
method), 119
write_tree_controller_meta_data() (*raf-*
con.gui.models.auto_backup.AutoBackupModel
method), 222
write_dict_to_json() (*in* *module* *raf-*
con.utils.storage_utils), 279
write_file() (*in module rafcon.utils.filesystem*), 273
write_output_data() (*raf-*
con.core.states.container_state.ContainerState
method), 139

V

value (*rafcon.core.state_elements.scope.ScopedData*
property), 125
VALUE_AS_STRING_STORAGE_ID (*raf-*
con.gui.controllers.global_variable_manager.GlobalVariableManagerController
attribute), 181
value_type (*rafcon.core.state_elements.scope.ScopedData*
property), 125
version (*rafcon.core.state_machine.StateMachine*
attribute), 163