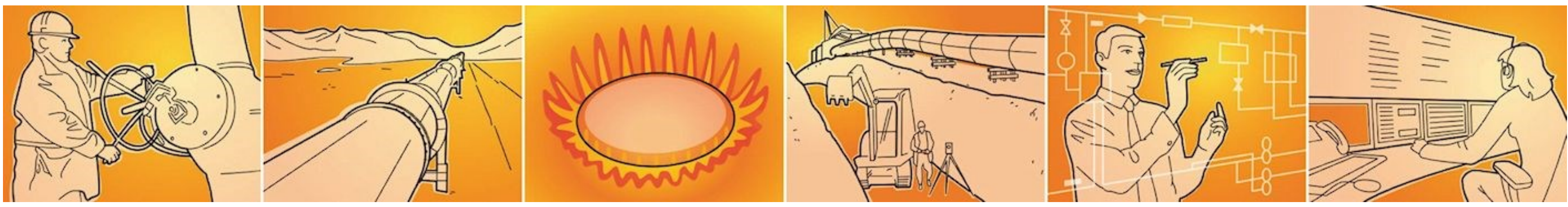
SIMONE API

Interface documentation

November 2023

Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group. All rights reserved.



Contents

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

1 SIMONE API

The API described in this document allows external programmers to communicate with SIMONE. Input data for simulations with SIMONE can be generated, scenarios can be calculated and results from a calculation with SIMONE can be read.

The API requires separate licensing for reading input and writing output. An additional license is required for calculating scenarios with simone_execute. Further functionality as required to manage network models and data allocation from within custom software as well as accessing special functions like state transfer, as it may be necessary for more integrated solutions, is provided under separate additional licensing and described in a separate document.

2 SIMONE Data

All data in SIMONE are addressed with a simple and direct concept, All objects get names and for all types of objects like nodes, pipes, valves etc. the relevant data are distinguished by shortcuts which are used as extensions to the names of the objects.

Each data point can be identified with the syntax <name of object>.<extension>.
Most objects used with SIMONE are network objects (elements and their nodes), which are generated with the network editor. But also all other objects (e. g. compressor units) are addressed using the above mentioned principle.

The user manual lists all possible input data (‘parameters’) and output data (‘variables’) with shortcuts, description texts and explanations.
The list is structured according to the types of objects and the usage of the data. Scenario parameters are input data for a simulation, variables are data read from calculated simulations.

Which parameters are necessary for a complete definition of a scenario to be created depends on the type of the scenario (type of simulation) and on the special task. For further information refer to the SIMONE Documentation.

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

3 Interface concepts

SIMONE data are managed in the hierarchy network - scenario. Following this hierarchy first a network is selected and then a scenario is opened. All reading or writing then applies to this current scenario. Only one scenario can be open at a time. For reading a scenario or updating a scenario, network and scenario must exist. For writing a scenario, the network must exist and a new scenario is created. The properties for the scenario and the times for start and end of the scenario must be set.
The interfaces use internal identifiers for the names of objects (obj_id) and extensions (ext_id). These identifiers are made available by a routine for the translation of names (*simone_varid*).
The engineering units to be applied for numerical data can be set to defaults (*simone_set_simulation_defaults*) or explicitly be defined. The interfaces use internal unit descriptors for this, which consist of a unit type and a unit code. These unit descriptors are built from a unit type (e.g. pressure, flow …) and respective abbreviations (like bar, m3/h, etc., see *simone_unit2des*, *simone_des2unit*).

Times are handled in ANSI C format (time_t) including date and time. The value „0“ is used to indicate „without time stamp“.

NOTE: On Windows only the 32-bit version of the API DLL uses a 32-bit value for *time_t* (maintaining compatibility to older versions) - see the introductory remarks of the section ‘time handling’ for more detail.

All interface routines return a status with success or error.

All variables can be read with the API and all scenario parameters can be written with the API. As with the interactive SIMONE user interface, the values of scenario parameters as valid at a given time can also be read back.

To provide all possibilities to create scenarios with the API like with the user interface, also interface routines to handle functions and load profiles are provided.

For reading variables any time within the definition of the simulation can be requested. If SIMONE did not calculate a value for this time, an interpolated value depending on the type of the variable is returned.

The exact interface and prototypes for all functions are defined in the simone_api.h header file and in the simone_api.bas, simone_api.net.

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

4 API Applications

The API functionality is implemented in a shared library (a DLL on Windows) and can be used from any application having access to the disk structure of a SIMONE installation[1]. The application may be either invoked from within SIMONE, i.e. it may run under the control of the SIMONE user interface or of the online cyclic control, or the application may be run independently and hence establish an instance of its own, also counted like another user from the licensing point of view.

To control access of multiple users (or instances) to the SIMONE data, a locking mechanism is used, that ensures only one instance to have access to a scenario at a time as well as inhibiting a network being updated while it is in use by somebody else[2].

As an API application may create a concurrent instance even in a single-user installation, the locking of resources is always active. In a multi-user installation, also user (access) rights control is active. But though an API application is always running in a user context, no (user) access right checking is done in the API. This means an API application always may access any SIMONE network or scenario that is not currently locked.

For each instance an own environment (configuration) is used. The SIMONE user-interface automatically creates such a configuration from a master template, using the username and computer name (Windows) or display name (UNIX) to create a unique identifier. This is being re-used any time the same user starts SIMONE on the same computer.

To attach or create the necessary environment to an API application, the *simone_init* or *simone_init_ex* function must be called first. If the application is invoked by SIMONE, the function should be called without specifying a configuration, thus make the application inherit the current environment, and allow to use the current network and scenario as defined in the user-interface or by the online control. This is the typical case for a SCADA interface program[3].

If an API application is running independently, a particular configuration (or a temporary copy, see *simone_init_ex)* should be attached, e.g. a copy of the master template (_SMASTER.INI on Windows, simenv.dir, simenv.pag on UNIX in the sys-directory of the installation) As in this case the API application constitutes an own instance, counted against the maximum number of users licensed, care may need to be taken about which license is being used. The routine *simone_init_setopt* allows to set the desired license number from the application. See the routine descriptions for details.

Note:
Because the parameter options –i and –c are internally used (i.e. interpreted) by the SIMONE API DLL, these option letters cannot be used any more for other purposes by the application.

4.1 API SDK

If the SIMONE-API option was selected during installation (what is the default if a license key was used with active SIMONE-API Option), a folder **SIMONE-API-SDK** is placed in the SIMONE installation directory with several subfolders for different language bindings.
C/C++ programmers should include the header file(s) simone_api.h and, optionally, simone_api_topo.h from inside the folder **SIMONE-API-SDK\C-C++**. . Since version 6.1 different installations for a 32-bit and 64-bit SIMONE are provided. Hence during link time the the matching import library has to be specifiedSince version 6.2 both variants are available in different subfolders,

[1] A special remote library is available under additional license that provides the API functionality also across the network.

[2] See the chapter „Multi-User Operations“ in the Userguide Supplements for a more detailed discussion of this topic.

[3] Please refer to the document „SIMONE Data Exchange using SIMONE API and Online Environment“ for more details.

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

i.e. 32-Bit programs the file:

SIMONE-API-SDK\C-C++\VS9lib\x86\simone_api.lib

and for a 64-Bit program:

SIMONE-API-SDK\C-C++\VS9lib\x64\simone_api.lib

needs to be linked.

NOTE: When using a 32-bit version, you may need take care on time_t handling, see the section about time handling.

The sub-folder Java contains the file **SimoneApi.jar** which should be on the class path for Java programs using the SIMONE-API Java language binding. The folder **Javadoc** contains the SIMONE-API documentation in Javadoc format.
Please note that Java programs need to have the files SimoneApi.jar and SimEdit.jar on the Java classpath during runtime. Both files can be found in the **Jbin** subfolder of the SIMONE installation.

The folder **COM** contains the SIMONE-API COM Object implementation. There are several versions of the COM object available supporting different interfaces and SIMONE versions. Customers starting with the COM object should always use the latest version of the COM object. The use of the SIMONE-API COM Object is described in a separate manual.

The folders VB.Net and VisualBasic contains the necessary files to use the SIMONE-API from VB-Net or Visual Basic respective Office programs like Excel (please note that a 64-bit SIMONE also requires a 64-bit Excel).

The library SimoneApiLua.dll required when using the LUA script language is to be found in exe directory of the SIMONE installation.

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

5 API Initialization and Scenario Handling

Before using the API, it must be initialized using *simone_init* or *simone_init_ex* and work should be terminated by *simone_end*.
All reading and writing applies to a current scenario. A scenario belongs to a network that must already exist and can be selected by *simone_select*.
The scenario then needs to be created or opened by *simone_open*.
Properties of the scenario can be set or changed by *simone_set_properties*, properties of an existing scenario can be read with the *simone_get_properties.*
The comment text as shown in the user-interface can be managed by *simone_set_scenario_comment*.
Unlike with the user interface, in the API the time interval of a scenario is handled by separate interfaces to get and set the initial and terminal time. These are documented in the section *Time Handling*.
A scenario should be closed by *simone_close* as soon as it is no longer needed.
To remove an existing scenario or delete it's results, use *simone_remove*.

5.1 simone_api_version()

Returns the version of the API being installed. This is the only routine besides *simone_init_setopt* (see below) that may be called without simone_init having been called before.

5.2 simone_init(init_file)

Initialize the API. This routine MUST be called before any other function of the API can be used (except *simone_api_version()* or the *simone_init_setopt()*).
At this time, a configuration file is attached. If not specified (as *init_file*), the configuration set by the environment is used. If the application was invoked by SIMONE, the configuration is inherited from the respective instance. Otherwise, an attempt is made to locate the configuration using the installation identifier provided on the command line with the –c switch (Windows only), or to use the configuration file pointed to by the –i switch (Windows only) or the environment variable SIMINI (Unix only).
If required, a dedicated SIMONE configuration can be set explicitly.

CAUTION:
It is the responsibility of the API user, to make sure not to attach a configuration that may be used by another instance at the same time (except when being called by this instance).
For convenience, *simone_init_ex* can be used, that allows to create and use a temporary private copy of a configuration.

Based on the configuration attached, also the license is being checked, i.e. whether or not the used SIMONE license allows for using the API. The license used is defined in the configuration.
If necessary, *simone_init_setopt* optionally allows to select a license to be configured in a temporary private copy. Then *simone_init_ex* .must be used instead.

Note:
If options are set to control the init – process with *simone_init_setopt()* function always a temporary copy of the configuration is used with the simone_init() function, which is deleted upon exiting the API application.

Optional parameter:

init_file	- specification of SIMONE-configuration file name of file without extension empty string or NULL: use configuration as from environment	IN

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

Return values:

simone_status_nolicense	no license requested license set with *simone_init_setopt*() not allowed
Simone_status_insuff_license	insufficient license level
Simone_status_badseq	API is already initialized
Simone_status_nofile	configuration not found
simone_status_ok	ok

5.3 simone_init_ex(init_file, flags)

This interface extends the functionality of *simone_init* according to flags.
If flags is set to SIMONE_FLAG_TMP_CONFIG, the configuration file specified as *init_file* or determined as described above for simone_init, is not used directly, a temporary private copy of this is created and used. This is the recommended method to ensure not two instances of an API application or the application and an instance of the user interface are working on the same configuration at the same time.

The temporary copy is deleted upon exiting the API application.

Optional parameters:

init_file	- specification of SIMONE-configuration file name of file without extension empty string or NULL: use configuration as from environment		IN
flags	SIMONE_NO_FLAG	use SIMONE-configuration, as with *simone_init*	IN
	SIMONE_FLAG_TMP_CONFIG	use temporary copy of configuration	

Return values:

simone_status_nolicense	no license requested license set with *simone_init_setopt*() not allowed
simone_status_insuff_license	insufficient license level
simone_status_badseq	API is already initialized
simone_status_nofile	configuration not found or error with creation of temporary copy of configuration
simone_status_ok	Ok

5.4 simone_init_setopt(initopt, optval)

Options to control the init - process for the SIMONE API are set with this function. These options must be set before one of the *simone_init()* or *simone_init_ex()* functions are called.

The options currently supported are

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

- SIMONE_INITOPT_REQUEST_LICENSE - setting the license to be used by the application.Licenses are identified by their number (normally referred to as CUSTID), which can be specified directly or by referring to the configured foreground, background or viewer license.

- SIMONE_INITOPT_SET_LICENSE_SERVER Set the license server to be used explicitly..The value optval for this option can be a list of hostnames or IP-adresses separated by ','

- SIMONE_INITOPT_SET_LS_CONNECT_TIMEOUT Define a timeout [milliseconds] for connecting to a SIMONE License Server, will be ignored if local dongle checking is active

Note:
If options are set to control the init – process, always a temporary copy of the configuration is used with the *simone_init()* or *simone_init_ex()* functions, which is deleted upon exiting the API application.

Parameters:

initopt	- option to be set – expressed by a value as described above	IN
optval	- string containing a string value matching the requested option depending on initopt	IN

Special values valid for SIMONE_INITOPT_REQUEST_LICENSE

foreground	This keyword will force to use the configured FULL license. If no full license is configured the default standalone license is used.
viewer	This keyword will force to use the configured VIEWER license. If no viewer license is configured the default standalone license is used. .
Background	This keyword will force to use the configured background license for the online cycle.
string containing an integer number	This number is used during simone_init () or simone_init_ex() to request a license.
string containing a name of an entry in the configuration	This referred entry in the configuration must contain a number which is used during simone_init or simone_init_ex to request a license. For setting of entries into the configuration refer to the simone_get_config_item function in the configuration chapter.

Return values:

simone_status_badseq	function makes no sense, if called after simone_init(),simone_init_ex()
simone_status_badpar	unknown option supplied or supplied option value invalid
simone_status_ok	Ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

5.5 simone_get_license_status(license_server, master_slave_mode, life_time)

In a redundant system, additional ‘slave’ license servers may be installed (using ‘slave’ dongles – see also the documentation about SIMONE redundancy support).
This routine provides information about the Master/Slave status of a SIMONE license server, including the remaining survival life time of a slave server.

Parameters:

license_server	- Name or IP-address of license server empty string or NULL: use the currently configured license server	IN
master_slave_mode	- SIMONE_MASTER | SIMONE_SLAVE	OUT
life_time	- remaining survival Life time as fraction of the maximum time [0.0, ... 1.0] only delivered, if slave	OUT

Return values:

simone_status_badseq	SIMONE API not initialized
simone_status_nolicence	No license
simone_status_ok	Ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

5.6 simone_change_network_dir(network_path, flags)

Change current directory for SIMONE networks. If the specified network_path is not defined or is not a valid SIMONE network path, an error simone_status_nofile is returned, unless the flag SIMONE_FLAG_MAKE_NETWORK_DIR is set.. If the flag is set and the network path does not exist, it will be created and the current network directory changed accordingly.

Parameter:

network_path	- Path for SIMONE networks to be used	IN
flags	- SIMONE_FLAG_MAKE_NETWORK_DIR	IN

Return values:

simone_status_nolicense	no license
simone_status_badseq	SIMONE API not initialized
simone_status_badpar	invalid network_path
simone_status_nofile	network_path does not exist network_path is not a simone network path
simone_status_ok	ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

5.7 simone_network_list_start(flags)

This routine prepares for a sequence of subsequent repeated calls to *simone_network_list_next()* to enumerate the networks contained in the current network directory. Only those networks will be returned to which the user has at least read access.

Parameters:

flags	- flags reserved for future use	IN

Return values:

simone_status_nolicense	no license
simone_status_badseq	SIMONE API not initialized,
simone_status_not_found	no networks found
simone_status_ok	Ok

5.8 simone_network_list_next(network, network_len, flags)

This function returns the name of the next network.

Parameters:

network	- name of network	OUT
network_len	- maximum length for name of network	IN
flags	- flags reserved for future use	IN

Return values:

simone_status_nolicense	no license
simone_status_badpar	network parameter is NULL network_len too small
simone_status_badseq	SIMONE API not initialized
simone_status_not_found	no (more) networks (no valid network name returned)
simone_status_ok	Ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

5.9 simone_select(network)

The network to be used is selected This has to be defined before a scenario can be opened or created. If the network is not specified, the current network as set in the configuration file is used (e.g. as set before by the SIMONE user-interface).
The network is locked for concurrent use, i.e. other users may still access the network, but not update the network model. Consequently, a lock error will occur if such update is already in progress by somebody else (normally this is during an ‘Activate Changes’ at the user-interface).

Note:
The current directory is changed, after *simone_end* the current directory is being restored to what it was before *simone_init*.

Parameter:

network	- Name of SIMONE network to be used empty string or NULL: use current network from configuration file	IN

Return values:

simone_status_nolicense	no license
simone_status_badpar	name of network too long
simone_status_nofile	network does not exist
simone_status_locked	no write permission because another application currently updates the network
simone_status_incompatible	network is not compatible with the actual version of the SIMONE software
simone_status_ok	Ok

5.10 simone_deselect(flags)

The actually open network is closed.

Parameter:

Flags	- reserved for future use	IN

Return values:

simone_status_nolicense	no license
simone_status_ok	Ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

5.11 simone_open(scenario, mode_flag)

Open or create a scenario. A network must have been selected before.
The mode parameter controls whether a new scenario is created, or an existing scenario is opened for writing or reading.
Only one scenario can be open at a time. The scenario is attempted to be locked exclusively, i.e. no other instance may access it at the same time. The lock is being released upon calling simone_close.

Parameters:

scenario	name of scenario to be opened empty string or NULL: use actual scenario from configuration file		IN
mode_flag	SIMONE_MODE_CREATE:	create new scenario for writing	IN
	SIMONE_MODE_WRITE:	open scenario for writing	
	SIMONE_MODE_READ:	open scenario for reading	
	SIMONE_FLAG_VISIBLE:	the scenario created shall be visible at the SIMONE User Interface	
	SIMONE_FLAG_NO_OPT:	disable optimization for writing scenario	
	SIMONE_FLAG_FORCE_OVERWRITE:	enforce writing mode value or setpoint value	
	SIMONE_FLAG_ATTR2SCENARIO	write attribute values to scenario	

Only one of the modes SIMONE_MODE_CREATE, SIMONE_MODE_WRITE or SIMONE_MODE_READ **must** be set in the mode_flag parameter, the flags SIMONE_FLAG_VISIBLE or SIMONE_FLAG_NO_OPT may be added to the mode_flag parameter.

A scenario created by the API is not visible at the user-interface by default. To have the scenario created treated like any other scenario which is generated interactively, the flag SIMONE_FLAG_VISIBLE may be set together with SIMONE_MODE_CREATE.

When a scenario is saved to disk, unnecessary items (e. g.: repeated scenario parameters) are deleted from the scenario. This optimization is prevented, if the SIMONE_FLAG_NO_OPT flag is set.

The flag SIMONE_FLAG_FORCE_OVERWRITE controls some behavior relevant in online interfacing, refer to the chapter Writing scenario entries for details.

The flag SIMONE_FLAG_ATTR2SCENARIO enforces *simone_write(), simone_write_with_flag(), simone_write_ex()* and *simone_write_array()* to rwrite attribute values of numeric type as scenario parameters (like in the GUI). This effectively overwrites an attribute value locally for a scenario. If this flag is not set in *simone_open()* and not set in *simone_write…()*, attribute values are global and saved only once for the whole network (for all scenarios).

For a newly created scenario some properties need to be defined. *Simone_open* sets the properties according to what is currently defined in the configuration file (as e.g. set from a calling instance). The same is true for the start- and end time of the scenario.

If the properties or times are to be set explicitly, *simone_set_properties* and *simone_set_times* can be used.

After opening a scenario in READ mode, the time for data retrieval is set to the end time. Use *simone_set_rtime,* to get data for another time instant. In particular before calling *simone_get_entry*, the start time must be set explicitely, if it is to retrieve entries from the start.

Note:
The type of simulation should be set properly before starting to write scenario parameters.

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

Return values:

simone_status_nolicense	no license
simone_status_badseq	no network selected scenario to be created already exists
simone_status_badpar	name of scenario too long (another) scenario already open invalid mode
simone_status_nofile	scenario cannot be created or opened
simone_status_locked	scenario already locked by another instance
simone_status_ok	ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

5.12 simone_set_readmode(flags)

Allows to set optional read mode.

Parameters:

flags	-	read mode		IN
		SIMONE_FLAG_READ_EFF	Read values only from scenario result database	
		SIMONE_FLAG_READ_CUR	Read values from scenario if available and if no data is in result database	
		SIMONE_FLAG_READ_INIQ	Read Q value from initial state	

Return values:

simone_status_ok	always returns Ok

5.13 simone_set_properties(runtype, owner, inic_file)

If the current scenario is open for writing, properties may be changed.
The type of simulation (runtype) can only be set if the scenario is still empty, i.e. before the first parameter is written to the scenario.
By default, *simone_open* sets the owner for a newly create scenario to the current simone user defined in the configuration. If it is redefined here, the owner must be a known simone user. The owner only applies in a multi-user installation – see the respective chapter in the User guide Supplements for more details.
The initial conditions for the scenario are defined by setting *inic_file* to INIT or to the name of another scenario.

CAUTION:
Neither *owner* nor *inic_file* are checked here for being valid.

Optional parameters:

runtype	- type of scenario (type of simulation) or 0, if to remain unchanged	IN
owner	- owner of scenario or empty string or NULL, if to remain unchanged	IN
inic_file	- name of initial conditions or empty string or NULL, if to remain unchanged	IN

Return values:

simone_status_nolicense	no license
simone_status_nofile	scenario not open
simone_status_badseq	no network selected scenario only open for reading runtype can only be set for an empty scenario
simone_status_badpar	wrong type of scenario (type of simulation) name of owner too long

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

	name of inic_file too long
simone_status_ok	ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

5.14 simone_get_properties(runtype, owner, owner_len, inic_file, inic_file_len)

If the current scenario is open, properties may be read from the scenario as set with *simone_set_properties* or as set during creation of the scenario either with *simone_open* or with the SIMONE user interface.

CAUTION:
Nor *owner* nor *inic_file* are checked here for being valid.

Parameters:

runtype	- type of scenario (type of simulation)	OUT
owner	- owner of scenario	OUT
owner_len	- maximum length for owner (including terminating zero)	IN
inic_file	- name of initial conditions	OUT
inic_file_len	- maximum length for inic_file (including terminating zero)	IN

Return values:

simone_status_nolicense	no license
simone_status_nofile	scenario not open
simone_status_badseq	no network selected
simone_status_badpar	name of owner too short name of inic_file too short
simone_status_ok	ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

5.15 simone_set_inic(inic_file, inic_time, flags)

This interface allows to specify/change the initial conditions to be used for a scenario. As opposed to *simone_set_properties* it also checks if the specified file exists and returns its time stamp, i.e. the simulation time for which it was created.

Further special handling is provided in the context of an online network. If PRSIM is specified as inic_file, the *initime* of the scenario (previously set by *simone_set_times* or implicitly from the environment) is checked and a best matching version of a stored terminal state of PRSIM is searched for. This may be the most recent PRSIM state (if *initime* is close to current time) or one of the states saved every hour for the last 24 hours, or one of the states saved every midnight. If a state is found, a copy is attached to the scenario, so any future execution of it will start on this PRSIM state as initial conditions.
In a repeated call you may need to set SIMONE_FLAG:OVERWRITE to replace an already attached file by another version.

The scenario must be open for writing for the call to succeed.

Parameters:

inic_file	- name of initial conditions		IN
inic_time	- time stamp of initial conditions		OUT
flags	- flags	SIMONE_NO_FLAG | SIMONE_FLAG_OVERWRITE default: SIMONE_NO_FLAG	IN

Return values:

simone_status_nolicense	no license
simone_status_nofile	scenario not open
simone_status_badseq	no network selected scenario only open for reading
simone_status_badpar	invalid inic_file, invalid inic_time
simone_status_not_found	initial conditions not existing, but initial conditions are set in scenario
simone_status_ok	ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

5.16 simone_set_scenario_comment(comment)

A comment is set, changed or removed for a scenario.

Parameter:

comment	- string with comment empty string or NULL: comment is removed	OUT

Return values:

simone_status_nolicense	no license
simone_status_nofile	no scenario open
simone_status_badseq	no network selected scenario only open for reading
simone_status_ok	ok

5.17 simone_get_scenario_comment(comment, comment_len)

A scenario comment is read from a scenario as set with *simone_set_scenario_comment* or as set during creation of the scenario with the SIMONE user interface.

Parameter:

comment	- string with comment	OUT
comment_len	- maximum length for comment (including terminating zero)	IN

Return values:

simone_status_nolicense	no license
simone_status_nofile	no scenario open
simone_status_badseq	no network selected scenario only open for reading
simone_status_ok	ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

5.18 simone_set_ simulation_defaults()

For some SIMONE calculation options (e.g. quality tracking on/off, zet formula, etc.), defaults can be set in the dialogue „Tools/Settings/Simulation” at the user interface. The corresponding scenario parameters are then set accordingly in a scenario, when it is created with the user interface (if the settings differ from the internal defaults of the simulation).

This routine provides this functionality with the API, i.e. it sets the respective parameters for the current scenario according to the settings in the actual configuration.

Return values:

simone_status_nolicense	no license
simone_status_nofile	no scenario open
simone_status_badseq	no network selected scenario only open for reading
simone_status_ok	ok

5.19 simone_close()

The currently open scenario is closed. The lock held for the scenario is being released.

Return values:

simone_status_nolicense	no license
simone_status_nofile	scenario not open
simone_status_ok	ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

5.20 simone_remove(scenario, flag)

An existing scenario or calculated results of a scenario are removed.

Parameters:

scenario	- name of scenario to be modified		IN
flag	- SIMONE_FLAG_REMOVE_ALL:	scenario is removed completely	IN
	SIMONE_FLAG_REMOVE_RESULTS:	only results of executed scenarios are deleted	
	SIMONE_FLAG_REMOVE_CORE_RESULTS:	same as SIMONE_FLAG_REMOVE_RESULTS but keeps terminal state if possible	IN

Return values:

simone_status_nolicense	no license
simone_status_badseq	no network selected scenario not closed
simone_status_badpar	name of scenario too long scenario does not exist
simone_status_nofile	errors with removing files
simone_status_locked	scenario already locked by another instance
simone_status_ok	ok

5.21 simone_end()

Working with the API ends. Any open scenario is closed. Any open network is released. The current directory is reset to the path valid before calling *simone_select*.

Return values:

simone_status_nolicense	no license
simone_status_badseq	no simone_init() or simone_init_ex() called before
simone_status_ok	ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

5.22 simone_scenario_list_start(flags)

This routine starts a sequence of calls to get a list of all scenarios in the current network. With subsequent repeated calls to *simone_scenario_list_next()* the names of all scenarios are returned. If detailed information for a scenario is required, *simone_scenario_list_info()* called directly after *simone_scenario_list_next()* returns detailed information about a scenario.

Parameters:

flags	- flags reserved for future use	IN

Return values:

simone_status_nolicense	no license
simone_status_badseq	SIMONE API not initialized, no network selected
simone_status_not_found	ok, no scenarios in network
simone_status_ok	ok

5.23 simone_scenario_list_next(scenario, scenario_len, flags)

This function lists the name of the next scenario in the current network.

Parameters:

scenario	- name of scenario	OUT
scenario_len	- maximum length for name of scenario	IN
flags	- flags reserved for future use	IN

Return values:

simone_status_nolicense	no license
simone_status_badpar	scenario parameter is NULL scenario_len too small
simone_status_badseq	SIMONE API not initialized no network selected no call of simone_scenario_list_start() before
simone_status_not_found	ok, no more scenarios in network
simone_status_ok	ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

5.24 simone_scenario_list_info(runtype, initial_condition, initial_condition_len, initime, termtime, owner, owner_len, comment, comment_len, flags)

List detailed information for a scenario.

Parameters:

runtype	- type of scenario	OUT
initial_condition	- name of initial condition of scenario NULL: do not return initial_condition	OUT
initial_condition_len	- maximum length for initial_condition including terminating zero 0: do not return initial_condition	IN
initime	- start time of scenario	OUT
termtime	- end time of scenario	OUT
owner	- owner of scenario NULL: do not return owner	OUT
owner_len	- maximum length for owner including terminating zero 0: do not return owner	IN
comment	- scenario comment NULL: do not return comment	OUT
comment_len	- maximum length for comment including terminating zero 0: do not return comment	IN
flags	- flags reserved for future use	IN

Return values:

simone_status_nolicense	no license
simone_status_badpar	initial_condition_len too small owner_len too small comment_len too small
simone_status_badseq	SIMONE API not initialized no network selected no call of simone_scenario_list_start() before no call of simone_scenario_list_next () before
simone_status_ok	Ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

5.25 simone_scenario_info(scenario, runtype, inic_file, inic_len, initime, termtime, owner, owner_len, comment, comment_len, flags)

List detailed information for a scenario. The scenario must not be open.

Parameters:

scenario	- name of scenario	IN
runtype	- type of scenario	OUT
inic_file	- name of initial condition of scenario NULL: do not return initial_condition	OUT
inic_len	- maximum length for initial_condition including terminating zero 0: do not return initial_condition	IN
initime	- start time of scenario	OUT
termtime	- end time of scenario	OUT
owner	- owner of scenario NULL: do not return owner	OUT
owner_len	- maximum length for owner including terminating zero 0: do not return owner	IN
comment	- scenario comment NULL: do not return comment	OUT
comment_len	- maximum length for comment including terminating zero 0: do not return comment	IN
flags	- flags reserved for future use	IN

Return values:

simone_status_nolicense	no license
simone_status_badseq	SIMONE API not initialized no network selected
simone_status_nofile	scenario not existing
simone_status_badpar	initial_condition_len too small owner_len too small comment_len too small
simone_status_ok	Ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

5.26 simone_get_info(mode, info, info_len, flags)

Inquire info about the current environment. It allows to obtain explicit information about network, scenario, or network directory that might have been set by the calling SIMONE instance (e.g. user interface or online control).

Parameters:

mode	- type of info SIMONE_CONFIGURED_NETWORK SIMONE_CONFIGURED_SCENARIO SIMONE_CURRENT_NETWORK_DIR SIMONE_VERSION	IN
info	- string for required info	OUT
info_len	- maximum length for required info including terminating zero	IN
flags	- flags reserved for future use	IN

Return values:

simone_status_nolicense	no license
simone_status_badseq	SIMONE API not initialized
simone_status_badpar	invalid mode
simone_status_not_found	No required info available
simone_status_ok	Ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

6 Time Handling

Times in the API are handled in ANSI C format (time_t), including date and time. This is different from the user interface, where date and time are handled separately.

SIMONE always uses the starting date of a scenario as a reference, and internally uses times of parameters as hours since midnight of this date. If a time references a subsequent day, it is denoted as e.g. 1\10:15, where 1\ indicates that 10:15 refers to the next day.

Using the ANSI C time format, the API application will normally not need to be concerned about the SIMONE time. However, to enable supporting the SIMONE time format in case it may be required, conversion routines are provided, which are documented under *Time Conversion.*

Times are required to be provided for a scenario to define the calculation time interval (initime, termtime) as well as to indicate the time for a scenario parameter or profile entry.

The calculation time interval for a scenario may be set or retrieved by *simone_set_times* and *simone_get_times*, respectively. If an application is invoked by SIMONE, the requested time interval (as set in the user-interface or by the online control) can be requested by *simone_get_configured_times*.

IMPORTANT NOTE (32-bit version of the Windows version only):

The 32-bit API DLL uses a 32-bit value for *time_t* (maintaining compatibility to older versions), so you might need take care passing the proper size.
Microsoft C/C++ supports defining _USE_32BIT_TIME_T to instruct the compiler to use 32-bit *time_t* - which does the trick.
Otherwise the provided header file *simone_api.h* redefines the relevant parameters of the respective API interfaces to be of 32-bit size, allowing to properly communicating time_t values by using variables of the same size.
Macros _*SIMAPI*_TIME(t) and _*PTR*_SIMAPI_TIME(t) are defined in the header file to support proper handling of the 64-bit time values to and from the API parameter.

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

6.1 simone_set_times(initime, termtime)

The start- and end time for the calculation of the current scenario can be changed using this function. Initime and termtime are defined in ANSI C format, i.e. including date and time. The date contained in initime is considered to be the starting date of the scenario as shown at the user interface.
For times of parameters SIMONE internally uses hours since midnight from the starting date of the scenario. This means, that if the start-time of a scenario is e.g. defined as 1.1.2000, 8:00 and the time of a stored parameter is defined as 1.1.2000, 18:00, for the parameter is only stored the hour (18:00). If the starting date of the scenario is modified by calling *simone_set_times (*e.g. to 2.1.2000 10:00)*,* the times of the parameters already stored are then valid at the modified date (i.e. in the example it gets to be 2.1.2000 18:00 then).
Hence, as times of parameters are not touched, changing start- and end time by this call may render parameters in the scenario having times before initime or after termtime. (In the example above, a parameter originally stored for 1.1.2000 9:00 would be interpreted to be valid at 2.1.2000 9:00 after the change of initime and then be outside the interval).

Use *simone_set_times_with_flag* if times of parameters should keep their date or if parameters outside the redefined calculation interval should be removed.

Parameters:

initime	- start time (date and time)	IN
termtime	- end time (date and time)	IN

Return values:

simone_status_nolicense	no license
simone_status_nofile	no scenario open
simone_status_badseq	no network selected scenario only open for reading
simone_status_invtime	start time after end time
simone_status_ok	ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

6.2 simone_set_times_with_flag(initime, termtime, flags)

This interface extends the functionality of *simone_set_times* according to *flags*.
If flags are set to SIMONE_NO_FLAG, the effect is exactly *as for simone_set_times* described above.

If SIMONE_FLAG_SHIFT_TIME_RESPECT_DATE is set and the starting date of the scenario is modified, the internal times of the parameters are adjusted according to the new date, as to keep their absolute time stamp (date and time).
(If, e.g. the starting date is changed from 2.1.2000 to 1.1.2000, a parameter originally set for 2.1.2000 10:00 will keep this time stamp, after the change it's time will be shown at the user interface as 1\10:00)
If the starting date is changed to a later day, parameters may get before the new initime. In this case their time is set to no time, i.e. to be valid from the start.

If SIMONE_FLAG_REM_BEFORE_INITIME or SIMONE_FLAG_REM_AFTER_TERMTIME (or both) are set, flag(s), scenario parameters with times before *initime* or after *termtime* are removed.

Note:
Entries with time stamp set to 0 (notime, valid from the start) are kept.

If SIMONE_FLAG_SHIFT_TIME_RESPECT_DATE is set together with the SIMONE_FLAG_REM_BEFORE_INITIME or SIMONE_FLAG_REM_AFTER_TERMTIME flag(s), first the internal times are adjusted before entries are deleted as requested

Profile definition entries are not affected, i.e. nor their times are adjusted nor any entry is removed.

Parameters:

initime	- start time (date and time)	IN
termtime	- end time (date and time)	IN
flags	- SIMONE_FLAG_SHIFT_TIME_RESPECT_DATE maintain date of times of scenario parameters - SIMONE_FLAG_REM_BEFORE_INITIME remove all parameters in scenario with times before *initime* - SIMONE_FLAG_REM_AFTER_TERMTIME remove all parameters in scenario with times after *termtime*	IN

Return values:

simone_status_nolicense	no license
simone_status_nofile	no scenario open
simone_status_badseq	no network selected scenario only open for reading wrong type of open scenario
simone_status_invtime	start time after end time
simone_status_ok	ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

6.3 simone_get_configured_times(initime, termtime)

Get start- and end time from configuration, i.e. as set by the user-interface or online control. These configured times are the same as used as default, when a new scenario is created by *simone_open*. This function does not require a scenario to be open. It is provided for an API application to know about the time interval for which it may have been invoked. If a scenario is open, however, the time interval as set for the scenario is returned.

Parameters:

initime	- start time (date and time)	OUT
termtime	- end time (date and time)	OUT

Return values:

simone_status_nolicense	no license
simone_status_badseq	corresponding entries not (completely) set in configuration file
simone_status_invtime	start time (initime) after end time (termtime)
simone_status_ok	ok

6.4 simone_get_times(initime,termtime)

Start time and end time of the current scenario are returned. The time interval is read from the scenario definition and from the current database of calculated results (which may differ from the first in case the calculation was prematurely aborted). The minimum start and maximum endtime of both is returned.

Parameters:

initime	- start time (date and time)	OUT
termtime	- end time (date and time)	OUT

Return values:

simone_status_nolicense	no license
simone_status_badseq	no network selected
simone_status_nofile	no scenario open
simone_status_noval	error reading times
simone_status_ok	ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

7 Objects, Names and IDs

All scenario parameters and result variables are addressed by the name of an object and an 'extension' denoting the type of data referred to, e.g. "A.P" denoting the pressure at node A.
The API routines require the parameters or variables to be addressed by their internal *object_id* and *extension_id*. Therefore *simone_varid* is available, that provides the respective translation.

The concept of ids is also used internally for load profiles and functions, which can also be manipulated using the API. The respective routines are described in the section *Load profiles and functions*. Translation of names of load profiles and functions is supported by simone_varid if the extension .DEF is used. The identifiers for object and extension (obj_id, ext_id) of the constant profile is returned with the call of simone_varid("const.def", obj_id, ext_id). Using the resulting pair of ids, the values of profiles and functions may be retrieved as well.

In case an API application needs to work e.g. on all nodes or all pipes of a network, *simone_get_first_object* and *simone_get_next_object* are provided to allow retrieving the names of all such objects in a loop. Combining the names with the relevant extensions and using *simone_varid* then enables to get the necessary pair of *object_id* and *extension_id*.

Similarly, API interface functions are available to create and handle object sets. See the respective sections below.

Groups of entries in a scenario may be marked as belonging to a common 'source', e.g. if the contents of one scenario is included into another one, the included entries are marked by the name of the source scenario. Internally again the source name is handled by an id, which can be retrieved or defined by *simone_define_source_name*.

7.1 simone_varid(varnam, obj_id, ext_id)

Returns identifiers for object and extension (obj_id, ext_id) of a SIMONE parameter or SIMONE variable from its text format representation ("<name>.<extension>")

Parameters:

varnam	- SIMONE parameter or variable specifier <name>.<extension> for network objects <profile name>.DEF for load profiles <function name>.DEF for functions <object_set name>.DEF for object sets	IN
Obj_id	- Object-ID	OUT
Ext_id	- Extension-ID	OUT

Return values:

simone_status_nolicense	no license
simone_status_badseq	no network selected
simone_status_badpar	invalid or unknown varnam
simone_status_ok	Ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

7.2 simone_varid_array(n_names, varnames, flags, obj_ids, ext_ids, stats)

This interface provides the same functions as *simone_varid*, but for an array of names of SIMONE parameters or variables, thus reducing the overhead that may become relevant if a larger number of translations is required, in particular when working with the remote API.

Parameters:

n_names	- number of elements in the arrays (names, obj_ids, ext_ids, stats)	IN
varnames	- array with SIMONE varnames	IN
flags	- interpret names as string array or as csv string	IN

SIMONE_NO_FLAG	names is string array
SIMONE_FLAG_STRING	names is csv string with ";", " " or "\t" as separator

obj_ids	- array of resulting object-IDs	OUT
ext_ids	- array of resulting extension-IDs	OUT
stats	- array of status indicators for each variable or parameter specified in names	OUT

simone_status_badpar	invalid or unknown name
simone_status_ok	Ok

Return values:

simone_status_nolicense	no license
simone_status_badseq	no network selected
simone_status_badpar	one or more elements in varnames could not be translated
simone_status_ok	Ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

7.3 simone_varid_info(obj_id, ext_id, object_type, data_type, unit_type)

Information about a SIMONE parameter or SIMONE variable is returned.

Scenario parameters are set with the *simone_write...* functions, and values of variables or parameters specified are read with the *simone_read...* functions.

Parameters with data_type SIMONE_DATA_FLOAT can be set with one of the *simone_write*, *simone_write_with_flag* or *simone_write_ex* functions.
Float values of variables or parameters can be read either with the *simone_read* or *simone_read_str* function.

Parameters defining a configuration of a compressor station can be set either with the *simone_write_configuration* or with the *simone_write_ex* function using the value_str parameter, values of the configuration of compressor stations are read with the *simone_read_str* function.

For user-defined attributes all data_types are possible. User defined attributes are set either with the *simone_write_attr* or *simone_write_ex* function using the value_str parameter. Float values of user-defined attributes can be read with the *simone_read* function. Values of user-defined attributes with all data_types can be read with the *simone_read_str* function.

The unit_type is to be used with the *simone_unit2des*, *simone_set_api_default_unit* and *simone_get_api_default_unit* functions.

Parameters:

Obj_id	- Object-ID		IN
Ext_id	- Extension-ID		IN
object_type	- Type of object		OUT
	SIMONE_OBJTYPE_NS	supply node	
	SIMONE_OBJTYPE_NO	other node	
	SIMONE_OBJTYPE_PIPE	pipe	
	SIMONE_OBJTYPE_CS	compressor station	
	SIMONE_OBJTYPE_CV	control valve	
	SIMONE_OBJTYPE_VA	valve	
	SIMONE_OBJTYPE_MS	metering station	
	SIMONE_OBJTYPE_NRV	non return valve	
	SIMONE_OBJTYPE_RECP	storage	
	SIMONE_OBJTYPE_SE	short cut	
	SIMONE_OBJTYPE_RE	resistor	
	SIMONE_OBJTYPE_SUB	subsystem	
	SIMONE_OBJTYPE_UNIT	compressor unit	
	SIMONE_OBJTYPE_MIX	blending- or mixing station	
	SIMONE_OBJTYPE_SYS	dummy type for system parameters	
	SIMONE_UNKNOWN		

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

data_type	- Type of value		OUT
	SIMONE_DATA_INT		
	SIMONE_DATA_FLOAT		
	SIMONE_DATA_STRING		
	SIMONE_DATA_BOOL		
	SIMONE_DATA_DATETIME		
	SIMONE_UNKNOWN		
unit_type	- Type for unit		OUT
	SIMONE_UNIT_TYPE_P	pressure	
	SIMONE_UNIT_TYPE_Q	flow (standard, mass)	
	SIMONE_UNIT_TYPE_T	temperature	
	SIMONE_UNIT_TYPE_V	Velocity	
	SIMONE_UNIT_TYPE_CV	Calorific-value	
	SIMONE_UNIT_TYPE_PWR	power	
	SIMONE_UNIT_TYPE_RHO	Density	
	SIMONE_UNIT_TYPE_L	length	
	SIMONE_UNIT_TYPE_D	diameter	
	SIMONE_UNIT_TYPE_RR	roughness	
	SIMONE_UNIT_TYPE_VOL	Volume (geometric)	
	SIMONE_UNIT_TYPE_H	height above sea level	
	SIMONE_UNIT_TYPE_TIME	time	
	SIMONE_UNIT_TYPE_ENERGY	energy	
	SIMONE_UNIT_TYPE_HAD	had	
	SIMONE_UNIT_TYPE_QVOL	flow at operating conditions	
	SIMONE_UNIT_TYPE_AC	linepack (volume)	
	SIMONE_UNIT_TYPE_GWH	linepack (energy)	
	SIMONE_UNIT_TYPE_TRACE	Water content	
	SIMONE_UNKNOWN		

Return values:

simone_status_nolicense	no license
simone_status_badseq	no network selected
simone_status_badpar	wrong parameters
simone_status_invid	wrong object id
simone_status_ok	Ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

7.4 simone_varid_info_ex(obj_id, ext_id, object_type, data_type, unit_type, flags)

Like simone_varid_info, but returns additional information from a SIMONE parameter or SIMONE variable in 'flags'. The returned information describes the exi_id in detail.

Parameters:

objid	- Object-ID see simone_varid_info()	IN
ext_id	- Extension-ID see simone_varid_info()	IN
object_type	- type of object see simone_varid_info() OUT	
data_type	- type of value see simone_varid_info()	OUT
unit_type	- type for unit see simone_varid_info()	OUT
flags	- info for (Object-ID, Extension-ID)	OUT

```
SIMONE_EXTINFO_TEXT   has text representation
SIMONE_EXTINFO_RDF    can be set in rdf
SIMONE_EXTINFO_FUNC   extid is function
SIMONE_EXTINFO_USERFUNC     extid is user defined function
         (also SIMONE_EXTINFO_FUNC will be set)
SIMONE_EXTINFO_ATTR   extid is user defined attrribute
SIMONE_EXTINFO_QUAL   extid is quality parameter
SIMONE_EXTINFO_SQUAL        extid is supply quality
         (also SIMONE_EXTINFO_SQUAL will be set)
SIMONE_EXTINFO_LIMIT  extid is limit(if SIMONE_EXTINFO_HILIMIT
         not set, it is lower limit, both set is high limit)
SIMONE_EXTINFO_HILIMIT  extid is upper limit definition
         (SIMONE_EXTINFO_LIMIT set as well)
SIMONE_EXTINFO_PDELTA       extid is delta value(pressure drop...)
SIMONE_EXTINFO_NODEX        extid is applicable to node ...
```

Return values:

simone_status_nolicense	no license
simone_status_badseq	no network selected
simone_status_badpar	wrong parameters
simone_status_invid	wrong object id
simone_status_ok	Ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

7.5 simone_varid_ex(varnam, obj_id, ext_id, obj_type, flags)

Like simone_varid this interface returns objectId and extensionId for a SIMONE parameter or variable defined by its text representation (“<name>.<extension>”) supllied in varnam, but extends the information by adding the object type.
By setting flags to SIMONE_FLAG_NODE or SIMONE_FLAG_ELEMENT the routine can be forced to only translate for objects of the indicated class.
Settting SIMONE_FLAG_NO_LOGS suppresses error logs, which may be unwanted if the routine is used for checking the varnam string only.

Parameters:

varnam	- SIMONE parameter or variable specifier <name>.<extension> for network objects <profile name>.DEF for load profiles <function name>.DEF for functions <object_set name>.DEF for object sets		IN
obj_id	- Object-ID		OUT
ext_id	- Extension-ID		OUT
obj_type	- Type of object		OUT
flags	SIMONE_NO_FLAG	return obj_id,ext_id, obj_type for first matching object	IN
	SIMONE_FLAG_NODE	return obj_id,ext_id, obj_type, if obj_type is node	
	SIMONE_FLAG_ELEMENT	return obj_id,ext_id , obj_type, if obj_type is element	
	SIMONE_FLAG_NO_LOGS	do not produce error logs for invalid name	

Return values:

simone_status_nolicense	no license
simone_status_badpar	invalid or unknown varnam or found oject type does not match with flags
simone_status_ok	ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

7.6 simone_var2name(obj_id, ext_id, name, name_len)

Returns SIMONE variable or parameter name ("<name>.<extension>") from obj_id, ext_id.

Parameters:

obj_id	- Object-ID	IN
ext_id	- Extension-ID	IN
name	- SIMONE parameter or variable specifier	OUT
name_len	- maximum of length of name to be read including terminating zero	IN

Return values:

simone_status_nolicense	no license
simone_status_invid	invalid obj_id or invalid ext_id
simone_status_badpar	name == NULL, name_len too small
simone_status_ok	ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

7.7 simone_id2name(id, name, name_len, obj_type)

Returns name and type of a SIMONE object defined by its object identifier.
The function supports topological objects and non-topological ID's of source names, functions, load profiles, etc.

Parameters:

id	- Object-ID, Function-ID, Profile-ID or Source-ID	IN
name	- SIMONE name	OUT
name_len	- maximum length for SIMONE name	IN
object_type	- Type of object	OUT

SIMONE_OBJTYPE_NS	supply node
SIMONE_OBJTYPE_NO	other node
SIMONE_OBJTYPE_PIPE	pipe
SIMONE_OBJTYPE_CS	compressor station
SIMONE_OBJTYPE_CV	control valve
SIMONE_OBJTYPE_VA	valve
SIMONE_OBJTYPE_MS	metering station
SIMONE_OBJTYPE_NRV	non return valve
SIMONE_OBJTYPE_RECP	Storage element
SIMONE_OBJTYPE_SE	Joint
SIMONE_OBJTYPE_RE	resistor
SIMONE_OBJTYPE_SUB	subsystem
SIMONE_OBJTYPE_UNIT	compressor unit
SIMONE_OBJTYPE_MIX	blending station
SIMONE_OBJTYPE_SYS	system parameter
SIMONE_ OBJTYPE_FUNCTION	function
SIMONE_ OBJTYPE_PROFILE	load profile
SIMONE_ OBJTYPE_SOURCE	source name
SIMONE_UNKNOWN	

Return values:

simone_status_nolicense	no license
simone_status_badpar	invalid id, name_len too short
simone_status_ok	ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

7.8 simone_get_first_object(req_object_type, req_subsys_name, obj_id, object_name, object_name_len, object_type, subsys_name, subsys_name_len)

This function (in conjunction with *simone_get_next_object*) allows to retrieve the object_id's and names of all objects (or all objects of a particular type and/or subsystem) defined in the current network along with their object type and the name of the subsystem to which they belong.

Use this call to get the first of the objects and then successive calls to *simone_get_next_object* to get the remaining objects.

In the call to *simone_get_first_object* the requested object type and optionally the name of a subsystem may be specified to restrict the set of objects being returned accordingly.

Parameters:

req_object_type	- type of object	IN

SIMONE_OBJTYPE_NS	supply node
SIMONE_OBJTYPE_NO	other node
SIMONE_OBJTYPE_PIPE	pipe
SIMONE_OBJTYPE_CS	compressor station
SIMONE_OBJTYPE_CV	control valve
SIMONE_OBJTYPE_VA	valve
SIMONE_OBJTYPE_MS	metering station
SIMONE_OBJTYPE_NRV	non return valve
SIMONE_OBJTYPE_RECP	storage
SIMONE_OBJTYPE_SE	short cut
SIMONE_OBJTYPE_RE	resistor
SIMONE_OBJTYPE_SUB	subsystem
SIMONE_OBJTYPE_UNIT	compressor unit
SIMONE_OBJTYPE_MIX	blending- or mixing station
SIMONE_OBJTYPE_SYS	dummy type for system parameters
SIMONE_OBJTYPE_ALL	all object types

req_subsys_name	- name of sub system for requested objects or NULL or empty string: objects for all subsystems are retrieved	IN
Obj_id	- Object-ID of first matching object	OUT
object_name	- SIMONE object name, if object_name_len > 0	OUT
object_name_len	- max. length of SIMONE object name or 0: do not return object name	IN
object_type	- type of object	OUT
subsys_name	- name of subsystem, if subsys_name_len > 0	OUT
subsys_name_len	- max. length of name of subsystem or 0: do not return name of subsystem	IN

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

Return values:

simone_status_nolicense	no license
simone_status_badseq	no network selected
simone_status_badpar	object_name_len too short subsys_name_len too short unknown subsystem in req_subsys_name
simone_status_error	internal error
simone_status_not_found	no matching objects found
simone_status_ok	ok, a first matching object was returned

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

7.9 simone_get_next_object(obj_id, object_name, object_name_len, object_type, subsys_name, subsys_name_len)

Returns name, id, type and subsystem of next object in current network according to the settings of the previous call to *simone_get_first_object*.

If no more matching objects are found, *simone_status_not_found* is returned. In this call, no useful values are returned for the parameters.

Parameters:

Obj_id	- Object-ID of matching object	OUT
object_name	- SIMONE object name, if object_name_len > 0	OUT
object_name_len	- max. length of SIMONE object name or 0: do not return object name	IN
object_type	- type of object	OUT
subsys_name	- name of subsystem, if subsys_name_len > 0	OUT
subsys_name_len	- max. length of name of subsystem or 0: do not return name of subsystem	IN

Return values:

simone_status_nolicense	no license
simone_status_badseq	no network selected
simone_status_badpar	object_name_len too short subsys_name_len too short
simone_status_error	internal error
simone_status_not_found	no more matching objects found
simone_status_ok	ok, a matching object was returned

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

7.10 simone_get_object_info(obj_id, object_type, subsys_id, in_obj_id, out_obj_id, parent_obj_id)

Returns more detail about an object of particular use if that object is an topology element.In this case, in addition to typeand subsystem informatiom, the object ids of the start and end nodes of the element are returned. If the object is a unit in a compressor station, the id of the compressor station is returned as parent_id. If the object is part of a blending station, the id of the station returned as parent_id.

Parameters:

obj_id	- Object-ID	IN
object_type	- type of object	OUT

SIMONE_OBJTYPE_NS	supply node
SIMONE_OBJTYPE_NO	other node
SIMONE_OBJTYPE_PIPE	pipe
SIMONE_OBJTYPE_CS	compressor station
SIMONE_OBJTYPE_CV	control valve
SIMONE_OBJTYPE_VA	valve
SIMONE_OBJTYPE_MS	metering station
SIMONE_OBJTYPE_NRV	non return valve
SIMONE_OBJTYPE_RECP	storage
SIMONE_OBJTYPE_SE	short cut
SIMONE_OBJTYPE_RE	resistor
SIMONE_OBJTYPE_SUB	subsystem
SIMONE_OBJTYPE_UNIT	compressor unit
SIMONE_OBJTYPE_MIX	blending- or mixing station
SIMONE_OBJTYPE_SYS	dummy type for system parameters

subsys_id	- id of subsystem the object belongs to	OUT
in_obj_id	- id of of first node of element	OUT
out_obj_id	- id of second node of element	OUT
parent_obj_id	- id of parent object	OUT

Return values:

simone_status_nolicense	no license
simone_status_badseq	no network selected
simone_status_invid	Invalid object_id
simone_status_badpar	Invalid parameters
simone_status_ok	Ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

7.11 simone_define_source_name(name, src_id)

Source names are used to build groups of parameters within a scenario. If the requested source name does not exist up to this time, this function creates a new source name entry in the scenario and returns the src_id. If the source name already exists, the matching src_id is returned. If the name is already in use for another type of entry in the actual scenario, an error is returned, because names for local functions, load profiles, macros and source names must be unique.

Parameters:

name	- source name	IN
src_id	- ID of source name	OUT

Return values:

simone_status_nolicense	no license
simone_status_nofile	no scenario open
simone_status_badseq scenario only open for reading	no network selected
simone_status_badpar	name = NULL or empty name already in use for local functions, load profiles, macros, configuration names
simone_status_error	internal errors
simone_status_ok	ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

7.12 simone_get_next_id (req_type, id, result_type)

This function allows to enumerate the id's of source names, functions, load profiles, object sets, etc. similar to what *simone_get_next_object* does for network objects.
Set ID=0 to get the first id matching the request type mask and get further id's returned on successive calls. The names and types of the returned ID's can be inquired by the *simone_id2name*.

Parameters:

req_type	- Mask with type(s) for ID's to be requested		IN
	SIMONE_OBJTYPE_SOURCE	source name	
	SIMONE_OBJTYPE_FUNCTION	function	
	SIMONE_OBJTYPE_PROFILE	load profile	
	SIMONE_OBJTYPE_MACRO	macro	
	SIMONE_OBJTYPE_OBJ_SET	object set	
	SIMONE_OBJTYPE_ACTION	action	
	SIMONE_OBJTYPE_ALL	all types	
id	- ID		IN/OUT
result_type	- Type of ID		OUT
	SIMONE_OBJTYPE_SOURCE	source name	
	SIMONE_OBJTYPE_FUNCTION	function	
	SIMONE_OBJTYPE_PROFILE	load profile	
	SIMONE_OBJTYPE_MACRO	macro	
	SIMONE_OBJTYPE_OBJ_SET	object set	
	SIMONE_OBJTYPE_ACTION	action	

Return values:

simone_status_nolicense	no license
simone_status_badseq	no network selected
simone_status_error	internal errors
simone_status_not_found	ok, no matching entry found
simone_status_ok	ok, matching entry found

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

7.13 simone_extid2name(ext_id, name, name_len)

This function returns the name of a SIMONE extension from ext_id.

Parameters:

ext_id	- Extension-ID	IN
name	- name of extension	OUT
name_len	- maximum length for name including terminating 0	IN

Return values:

simone_status_nolicense	no license
simone_status_badseq	no network selected
simone_status_invid	invalid extension id
simone_status_badpar	invalid parameter name = NULL, invalid parameter name_len too small
simone_status_ok	ok

7.14 simone_extname2id(ext_name, ext_id, flags)

This function returns the identifier for an extension of a SIMONE parameter or SIMONE variable. By settting SIMONE_FLAG_NO_LOGS error logs can be suppressed in case the call is used for checking only.

Parameters:

ext_name	- Extension		IN
ext_id	- ID of extension		OUT
flags	- SIMONE_FLAG_NO_LOGS	do not produce error logs for invalid name of extension	IN

Return values:

simone_status_nolicense	no license
simone_status_badseq	no network selected
simone_status_badpar	name does not exist in SIMONE database
simone_status_ok	ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

8 Unit Handling

SIMONE offers a broad selection of engineering units available to input or output values of parameters and variables.

In the API, units are selected by unit descriptors consisting of a unit type and a unit code. *simone_unit2des* is provided to build a unit descriptor from the unit type and the standard text abbreviation of a particular unit. For valid values for the unit type refer to the header files (simone_api.h, simone_api.bas, simone_api.net), for valid text abbreviations refer to the unit tables at the end of this chapter. *simone_des2unit* allows to get the abbreviation text from a unit descriptor.

At the user interface, default units can be set that are used if no unit is specified for a particular input/output operation. These settings are stored in the respective configuration file of the instance. If an API application is using such a configuration (or a copy of it), the stored defaults will be used, if SIMONE_UNIT_DEFAULT is supplied as the unit descriptor.

In some applications, however, dedicated units might need to be specified (e.g. when interfacing to a SCADA system). In this case either the respective unit descriptors may be built using *simone_unit2des* and supplied to the input or output routines. Or, if always the same unit is to be used for a particular type, this may also be set as a default unit for the API using *simone_set_api_default_unit*. This default will be valid for the actual run of the application only, and unless it is redefined by another call to the function.

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

8.1 simone_unit2des(unit_type, unit_abbr, unit, flag)

Translate unit type and abbreviation to unit descriptor.

The unit_type of a scenario parameter or variable can be read with the *simone_varid_info* function.

Usual abbreviations as shown at the user interface are accepted. Because these abbreviations of units for output vary for different installed languages, it is strongly recommended to use the abbreviations of units for input when programming, which usually avoid special characters like C instead of °C or m3 instead of m³. Refer to the section with unit tables below for details.

Note:
The abbreviation must be enclosed in square brackets.

Parameters:

unit_type	- type of unit, refer to simone_api.h, simone_api.bas, simone_api.net	IN
unit_abbr	- text abbreviation of unit abbreviation must be in brackets, e. g. [km] empty string or NULL: create unit descriptor which only contains a unit_type	IN
unit	- unit descriptor	OUT
flag	- reserved	IN

Return values:

simone_status_nolicense	no license
simone_status_badpar	abbreviation not valid
simone_status_invid	unit type not valid
simone_status_ok	ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

8.2 simone_des2unit (unit, unit_abbr, abbr_len, flag)

Get unit abbreviation from unit descriptor. Abbreviations of units for output vary for different installed languages from units for input, which usually avoid special characters like C instead of °C or m3 instead of m³. Refer to the section with unit tables below for details.

Parameters:

unit	- unit descriptor	IN
unit_abbr	- abbreviation of unit depending on flag abbreviation is returned in brackets, e. g. [km]	OUT
abbr_len	- maximum length for unit abbreviation	IN
flag	- select input – or output abbreviation SIMONE_FLAG_OUTPUT_ABBR (default) SIMONE_FLAG_INPUT_ABBR	IN

Return values:

simone_status_nolicense	no license
simone_status_badpar	abbr_len too small
simone_status_invid	unit not valid
simone_status_ok	ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

8.3 simone_set_api_default_unit(unit_type, unit)

Set default unit for specified unit type to be used by API input/output functions, if unit is set to SIMONE_UNIT_DEFAULT. This does not affect the settings for the user interface.
The unit_type of a scenario parameter or variable can be read with the *simone_varid_info* function.

Parameters:

unit_type	- type of unit, refer to simone_api.h, simone_api.bas, simone_api.net	IN
unit	- unit descriptor specifying the default unit	IN

Return values:

simone_status_nolicense	no license
simone_status_badpar	unit not valid
simone_status_invid	unit type not valid
simone_status_ok	ok

8.4 simone_set_api_default_unit_ex(unit_type, unit, flags)

This interface optionally extends the effect of the setting when the flag SIMONE_SET_IN_CONFIGURATION is set. In this case the default unit settings in the current environment of the API program are modified to make them effective not only for the parameters set or variables read with the API calls, but also for all other processes in this context. This may be important in particular when user defined functions are used in a simulation executed from within the actual program.
Note: If the API program is launched from the SIMONE GUI, the change of settings will persist afterwards for the current user.

Parameters:

unit_type	- type of unit, refer to simone_api.h, simone_api.bas, simone_api.net	IN
unit	- unit descriptor specifying the default unit	IN
Flags	SIMONE_NO_FLAG set default unit for API read/write functions SIMONE_SET_IN_CONFIGURATION set default unit also in configuration to be valid for all processes in the current environment	IN

Return values:

simone_status_nolicense	no license
simone_status_badpar	unit not valid
simone_status_invid	unit type not valid
simone_status_ok	Ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

8.5 simone_get_api_default_unit(unit_type, unit)

Get default unit currently valid in the API for a particular unit type. If no default unit has been explicitly set before (by a previous call to *simone_set_api_default_unit*), the default from the configuration set by *simone_init* is returned.
The unit_type of a scenario parameter or variable can be read with the *simone_varid_info* function.

Parameters:

unit_type	- type of unit, refer to simone_api.h, simone_api.bas, simone_api.net	IN
unit	- unit descriptor of the default unit	OUT

Return values:

simone_status_nolicense	no license
simone_status_invid	unit type not valid
simone_status_ok	ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

8.6 Unit tables

The following tables list engineering units available to input or output values of parameters and variables.
For each unit it is also listed whether it is considered to be a metric or imperial unit. This is respected, if a unit conversion involves a (standard) volume of gas. Then the metric unit is considered to refer to the metric reference conditions selected for the network, whereas the imperial unit is taken as described by the imperial reference conditions selected. Please note that the reference conditions can only be set once in the network editor. See the userguide for further details.

Note:
The output abbreviations depend on the chosen language used by the SIMONE GUI. The abbreviations for output of the units in the following tables are valid for the English version.
The abbreviations of units for input are fixed for all chosen languages. So it is strongly recommended to use the latter when programming.

8.6.1 Pressure units

Name of unit	Abbreviation of unit for output	Abbreviation of unit for input	Unit system
bar	bar	bar	metric
MPa	MPa	MPa	metric
barg	barg	barg	metric
MPag	MPag	MPag	metric
kPag	kPag	kPag	metric
at	at	at	metric
ata	ata	ata	metric
atm	atm	atm	metric
kPah	kPah	kPah	metric
kPa	kPa	kPa	metric
barh	barh	barh	metric
psia	psia	psia	imperial
psig	psig	psig	imperial
Pa	Pa	Pa	metric

Use SIMONE_UNIT_TYPE_P as unit type parameter.

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

8.6.2 Flow units

Name of unit	Abbreviation of unit for output	Abbreviation of unit for input	Unit system
mil.Nm³/d	mil. Nm³/d	mil.Nm3/d	metric
1000Nm³/h	1000 Nm³/h	1000Nm3/h	metric
Nm³/s	Nm³/s	Nm3/s	metric
Nm³/h	Nm³/h	Nm3/h	metric
Nm³/d	Nm³/d	Nm3/d	metric
kilowatt	kW	kW	metric
megawatt	MW	MW	metric
kilogram/second	kg/s	kg/s	metric
1000Nm³/d	1000 Nm³/d	1000Nm3/d	metric
mmscfd	mmscfd	mmscfd	imperial
mmscfh	mmscfh	mmscfh	imperial
mscfh	mscfh	mscfh	imperial
scfm	scfm	scfm	imperial
scfs	scfs	scfs	imperial
pounds per second	lb/s	lb/s	imperial

Use SIMONE_UNIT_TYPE_Q as unit type parameter.

8.6.3 Units for calorific value

Name of unit	Abbreviation of unit for output	Abbreviation of unit for input	Unit system
kilowatthours/m³	kWh/m³	kWh/m3	metric
megajoule/m³	MJ/m³	MJ/m3	metric
BTU/cu ft	BTU/scf	BTU/scf	imperial

Use SIMONE_UNIT_TYPE_CV as unit type parameter.

8.6.4 Temperature units

Name of unit	Abbreviation of unit for output	Abbreviation of unit for input	Unit system
Kelvin	°K	K	metric
°Celsius	°C	C	metric
Fahrenheit	°F	F	imperial
Rankine	Ra	Ra	imperial

Use SIMONE_UNIT_TYPE_T as unit type parameter.

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

8.6.5 Linepack units

Name of unit	Abbreviation of unit for output	Abbreviation of unit for input	Unit system
mil.Kg	mil.kg	mil.kg	metric
mil.Nm³	mil.Nm³	mil.Nm3	metric
1000 Nm³	1000 Nm³	1000Nm3	metric
Nm³	Nm³	Nm3	metric
mmscf	mmscf	mmscf	imperial
mscf	mscf	mscf	imperial
scf	scf	scf	imperial
mmlb	mmlb	mmlb	imperial

Use SIMONE_UNIT_TYPE_AC as unit type parameter.

8.6.6 Units for inventory energy

Name of unit	Abbreviation of unit for output	Abbreviation of unit for input	Unit system
MWh	MWh	MWh	metric
GWh	GWh	GWh	metric
GJ	GJ	GJ	metric
MMBtu	MMBtu	MMBtu	imperial
BBtu	BBtu	BBtu	Imperial

Use SIMONE_UNIT_TYPE_GWH as unit type parameter.

8.6.7 Units for water content

Name of unit	Abbreviation of unit for output	Abbreviation of unit for input	Unit system
g/m³	g/m³	g/m3	metric
lb/scf	lb/scf	lb/scf	imperial
lb/MMscf	lb/MMscf	lb/MMscf	imperial

Use SIMONE_UNIT_TYPE_TRACE as unit type parameter.

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

8.6.8 Power units

Name of unit	Abbreviation of unit for output	Abbreviation of unit for input	Unit system
kilowatts	kW	kW	metric
megawatts	MW	MW	metric
gigawatts	GW	GW	metric
PS	PS	PS	metric
horsepower	hp	hp	imperial

Use SIMONE_UNIT_TYPE_PWR as unit type parameter.

8.6.9 Velocity units

Name of unit	Abbreviation of unit for output	Abbreviation of unit for input	Unit system
meter/second	m/s	m/s	metric
kilometer/hour	km/h	km/h	metric
feet/second	ft/s	ft/s	imperial
miles/hour	mph	mph	imperial

Use SIMONE_UNIT_TYPE_V as unit type parameter.

8.6.10 Length units

Name of unit	Abbreviation of unit for output	Abbreviation of unit for input	Unit system
kilometer	km	km	metric
meter	m	m	metric
mile	mi	mi	imperial

Use SIMONE_UNIT_TYPE_L as unit type parameter.

8.6.11 Diameter units

Name of unit	Abbreviation of unit for output	Abbreviation of unit for input	Unit system
meter	m	m	metric
millimeter	mm	mm	metric
centimeter	cm	cm	metric
inch	in	in	imperial

Use SIMONE_UNIT_TYPE_D as unit type parameter.

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

8.6.12 Units for height above sea level

Name of unit	Abbreviation of unit for output	Abbreviation of unit for input	Unit system
meter	m	m	metric
feet	ft	ft	imperial

Use SIMONE_UNIT_TYPE_H as unit type parameter.

8.6.13 Roughness units

Name of unit	Abbreviation of unit for output	Abbreviation of unit for input	Unit system
millimeter	mm	mm	metric
micrometer	µm	um	metric
inch	in	in	imperial
micro inches	µin	min	imperial

Use SIMONE_UNIT_TYPE_RR as unit type parameter.

8.6.14 Density units

Name of unit	Abbreviation of unit for output	Abbreviation of unit for input	Unit system	
1	1	1		= relative density
kilograms/cubic meter	kg/m³	kg/m3	metric	Base density
pounds per cubic feet	lb/ft³	lb/ft3	imperial	Base density

Use SIMONE_UNIT_TYPE_RHO as unit type parameter.

8.6.15 HAD units

Name of unit	Abbreviation of unit for output	Abbreviation of unit for input	Unit system
kJ/kg	kJ/kg	kJ/kg	metric, imperial
m	m	m	metric
km	km	km	metric
ft	ft	ft	imperial

Use SIMONE_UNIT_TYPE_HAD as unit type parameter.

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

8.6.16 Energy units

Name of unit	Abbreviation of unit for output	Abbreviation of unit for input	Unit system
kWh	kWh	kWh	metric, imperial
kJ	kJ	kJ	metric, imperial
Btu	Btu	Btu	metric, imperial

Use SIMONE_UNIT_TYPE_ENERGY as unit type parameter.

8.6.17 Units for flow at operating conditions

Name of unit	Abbreviation of unit for output	Abbreviation of unit for input	Unit system
m3/s	m3/s	m3/s	metric, imperial
m3/min	m3/min	m3/min	metric
m3/h	m3/h	m3/h	metric
1000m3/h	1000m3/h	1000m3/h	metric
cft/s	cft/s	cft/s	imperial
cft/min	cft/min	cft/min	imperial
cft/h	cft/h	cft/h	imperial

Use SIMONE_UNIT_TYPE_QVOL as unit type parameter.

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

8.6.18 Time units

Name of unit	Abbreviation of unit for output	Abbreviation of unit for input	Unit system	
Seconds	sec	dt_s	metric, imperial	Seconds since 1.1.1970 00:00 UTC 1082959115
Date	date	dt_ds	metric, imperial	26.04.2004
DateTime	datetime	dt_dt	metric, imperial	26.04.2004 07:58:35
Time	time	dt_t	metric, imperial	07:58:35
USDate	usdate	dt_uds	imperial	04/26/2004
USDateTime	usdatetime	dt_usdt	imperial	04/26/2004 07:58:35 AM
USTime	ustime	dt_ust	imperial	07:58:35 AM
LocDate	ldate	dt_lds		Date appropriate to current locale
LocDateTime	ldatetime	dt_ldt		Date and time appropriate to current locale
LocTime	ltime	dt_lt		Time appropriate to current locale
DTDay	dtday	dt_day	metric, imperial	26
DTMon	dtmon	dt_mon	metric, imperial	04
DTYear	dtyear	dt_year	metric, imperial	2004

Use SIMONE_UNIT_TYPE_TIME as unit type parameter.

Note:
The time units are used for user-defined attributes of the type SIMONE_DATA_DATETIME. User-defined attributes are set with the simone_write_ex function and they are read with the simone_read_str function.

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

9 Reading Data

Reading applies to the current scenario, which must be open in READ mode.

Results of scenarios are always stored in time slots, even if it is for steady state calculations. SIMONE variables are hence retrieved by time, which must be set by *simone_set_rtime* before reading. The time stamps of the time slots depend on the time step control that was used in the calculation of the scenario, which may have lead to irregular intervals (there is a basic time step interval, but additional steps are introduced automatically whenever control parameters or boundary values change in the input).

If data for all time slots calculated shall be retrieved, first calling *simone_set_rtime* with a zero time specification (i.e. no time) and then using *simone_next_rtime*, allows to retrieve the time stamps of all slots.

If data are required for dedicated times (e.g. at regular intervals), these times can be set by *simone_set_rtime* as well. In this case, the read interfaces will return appropriate values according to the type of data requested, e.g. interpolated values for pressures or the state of a valve valid for the given time, as in the user interface.

Most SIMONE data are represented as float values. There are a few exceptions however, like control modes, (compressor station) configurations, etc. Only float values are dealt with at the API interfaces in a binary (float) form. All others are communicated as character strings.
The data_type of a variable can be inquired with the *simone_varid_info* function. Depending on the data_type different interface functions can be used:
simone_read supports float values only. To read other data types, *simone_read_str* must be used. For convenience, *simone_read_str* optionally allows to get float values as formatted strings.
In particular user-defined attributes may use different data_types. See the description *simone_write_attr* for details on their text format representation.

The read interfaces are primarily intended to read calculated results, but also values of scenario parameters (as valid at the requested time) and values of user-defined attributes are returned if requested.
For special applications *a simone_get_entry* function is provided, that allows to read back the scenario entries rather than results data.

If calculated values are to be retrieved, it is recommended to first use *simone_calculation_status* to figure out if the scenario has been yet executed successfully.

If the execution was successful, all result values will be available for any time within the calculation interval, except for the flowrate through a valve, which is only calculated by SIMONE, if the valve is being switched during the calculation. If parameters like setpoints or metered pressures are read back with *simone_read*, they may be not available for all or part of the time interval. In such a case, a status SIMONE_STATUS_NOVAL will be returned.

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

9.1 simone_set_rtime(rtime)

The time for subsequent calls to *simone_read*, *simone_read_str* or *simone_get_entry* is set. If no time is given (rtime = 0), the time for data retrieval is set to the first time slot of the scenario.

Parameter:

rtime	-	retrieval time, next call to *simone_read* will return values for this time. next call to *simone_get_entry* will return entry for this time	IN

Return values:

simone_status_nolicense	no license
simone_status_nofile	no scenario open
simone_status_badseq	no network selected, scenario not open in READ mode
simone_status_ok	ok

9.2 simone_get_rtime(rtime)

The time set for subsequent calls to *simone_read*, *simone_read_str* or *simone_get_entry* is returned.

Parameter:

rtime	-	set retrieval time, next call to *simone_read* will return values for this time. next call to *simone_get_entry* will return entry for this time	OUT

Return values:

simone_status_nolicense	no license
simone_status_nofile	no scenario open
simone_status_badseq	no network selected, scenario not open in READ mode
simone_status_ok	ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

9.3 simone_next_rtime(rtime)

The time stamp of the next time slot after the last time set by the function *simone_set_rtime* is returned.

Parameter:

rtime	- next retrieval time after actual retrieval time	OUT

Return values:

simone_status_nolicense	no license
simone_status_nofile	no scenario open
simone_status_badseq	no network selected, scenario not open in READ mode
simone_status_invtime	no next retrieval time after actual retrieval time, actual retrieval time is last time with calculated values
simone_status_ok	ok

9.4 simone_set_next_rtime(rtime)

The time stamp of the next time slot after the last time set by the function *simone_set_rtime* is returned and set for subsequent calls to *simone_read*, *simone_read_str*.

Parameter:

rtime	- actual retrieval time 0: set first retrieval time	IN
	- next retrieval time after actual retrieval time for which SIMONE calculated values	OUT

Return values:

simone_status_nolicense	no license
simone_status_nofile	no scenario open
simone_status_badseq	no network selected, scenario not open in READ mode
simone_status_invtime	no next retrieval time after actual retrieval time, actual retrieval time is last time with calculated values
simone_status_ok	ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

9.5 simone_read(obj_id, ext_id, unit, value)

The value of the variable or parameter specified is read for the actual retrieval time set with *simone_set_rtime*. Only such variables or parameters can be read, which can be represented as float values, otherwise SIMONE_STATUS_NOFLOAT is returned.
Use *simone_read_str* to read such values.

If no time step was calculated and stored for the actual retrieval time, the value returned is derived from the values calculated for adjacent time steps depending on the type of the variable, i.e. results like pressures and flows are linearly interpolated whereas e.g. setpoints are returned as valid at the actual retrieval time.

If no value is defined for the actual retrieval time (like e.g. a setpoint that has not been used), SIMONE_STATUS_NOVAL is returned.

A unit descriptor (or SIMONE_UNIT_DEFAULT) must be supplied to indicate in which unit the value is to be returned. Use *simone_unit2des* to generate a valid unit descriptor.
If SIMONE_UNIT_DEFAULT is supplied, the value will be returned in the unit set by *simone_set_api_default_unit* or as the default unit was set at the time of *simone_init* from the configuration.
For upward compatibility with earlier versions, also the symbolic names for units of flow or pressure as defined in simone_api.h, simone_api.bas, simone_api.net can be supplied instead of a unit descriptor.

Parameters:

obj_id	- object-ID of variable	IN
ext_id	- extension-ID of variable	IN
unit	- unit descriptor for requested value’s unit	IN
value	- value of variable	OUT

Return values:

simone_status_nolicense	no license
simone_status_nofile	no scenario open
simone_status_badseq	no network selected or scenario not open in READ mode
simone_status_badpar	invalid unit descriptor
simone_status_invid	obj_id or extension_id or combination of both invalid
simone_status_nofloat	value cannot be represented as a float
simone_status_noval	value not defined for the actual retrieval time
simone_status_ok	ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

9.6 simone_read_array(n_values, obj_ids, ext_ids, units, values, stats)

This interface provides the same function as *simone_read*, but for an array of SIMONE parameters or variables, thus reducing the overhead that may become relevant if a larger number of values is to be retrieved, in particular when working with the remote API.
The values of *n_values* variables or parameters as specified by obj_ids, ext_ids is read for the actual retrieval time and returned in an array. If all variables' values are successfully read, the function returns with the ok status.
If the license is not sufficient, no scenario is open or no network is selected, values cannot be read and an appropriate status is returned. In case there are problems with one or more elements of the list, a respective status is returned, and the stats array can be checked to identify which list element(s) have a problem.

Parameters:

n_values	- number of variables to be read	IN
obj_ids	- object-IDs of variables	IN
ext_ids	- extension-IDs of variables	IN
units	- unit descriptors for requested values' units	IN
values	- values of variables	OUT
stats	- array of status indicators for each variable read	OUT

simone_status_badpar	invalid unit descriptor
simone_status_invid	obj_id or extension_id or combination of both invalid
simone_status_nofloat	value cannot be represented as a float
simone_status_noval	value not defined for the actual retrieval time
simone_status_ok	ok

Return values:

simone_status_nolicense	no license, no value was read
simone_status_nofile	no scenario open, no value was read
simone_status_badseq	no network selected or scenario not open in READ mode, no value was read
simone_status_badpar	at least one invalid unit descriptor
simone_status_invid	at least one obj_id or extension_id or combination of both invalid
simone_status_nofloat	at least one value cannot be represented as a float
simone_status_noval	at least one value not defined for the actual retrieval time
simone_status_ok	ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

9.7 simone_read_str(obj_id, ext_id, unit, value, width, precision)

This interface is provided to allow also for reading of values that can only be represented as text strings.
If the requested variable or parameter is a float number, it is formatted (right aligned) according to the parameters *width* and *precision*.

All other conditions and behavior are as described for *simone_read*.
Parameters:

obj_id	- object-ID of variable	IN
ext_id	- extension-ID of variable	IN
unit	- unit descriptor for requested value's unit	IN
value	- formatted value of variable or string refer to *simone_write_attr, simone_write_configuration* for special formats	OUT
width	- maximum length for value string (including terminating zero)	IN
precision	- number of decimals, if float value is to be formatted	IN

Return values:

simone_status_nolicense	no license
simone_status_nofile	no scenario open
simone_status_badseq	no network selected scenario not open in READ mode
simone_status_badpar	invalid unit descriptor
simone_status_invid	*obj_id* or *extension_id* or combination of both invalid
simone_status_noval	value not defined for the actual retrieval time
simone_status_ok	Ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

9.8 simone_get_entry_set_filter(flag, value)

This function sets a filter for subsequent calls of *simone_get_entry()*, causing them to return only scenario entries matching the filter condition. Multiple calls to this routine can be used to establish multiple conditions.

The first parameter indicates the type of condition, the second the value to match – see table below.

If the filter is for an ID, the filter forces the respective id to match for an entry to be returned, if the filter flag specifies a type, the repecitve type must match. So e.g. the sequence

```
simone_get_entry_set_filter(SIMONE_FLAG_FILTER_OBJ_ID, <obj_id>)
simone_get_entry_set_filter(SIMONE_FLAG_FILTER_PARAM_TYPE,
        SIMONE_PARAM_TYPE_SUPPLY_OFFTAKE | SIMONE_PARAM_TYPE_QUALITY)
```

will make simone_get_entry() return all entries for the defined obj_id that define supply or offtake or quality

Note:
call *simone_get_entry_reset_filter()* to remove filters again to resume unfiltered operation

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

Parameters:

flag	- filter flag	IN
value	- filter value	IN

For combination of filter flag and filter value refer to the following table:

Filter flag	**Filter value**
SIMONE_FLAG_FILTER_OBJ_ID	valid Object-Id
SIMONE_FLAG_FILTER_EXT_ID	valid Extension-Id
SIMONE_FLAG_FILTER_COND_FLAGS	SIMONE_FLAG_IF SIMONE_FLAG_IF1 SIMONE_FLAG_IFN SIMONE_FLAG_IF1N SIMONE_FLAG_IF_ALL
SIMONE_FLAG_FILTER_COND_ID	valid Condition-Id
SIMONE_FLAG_FILTER_FUNC_ID	valid Function-Id valid Profile-id
SIMONE_FLAG_FILTER_SRC_ID	valid Source-Id
SIMONE_FLAG_FILTER_OBJ_TYPE	SIMONE_OBJTYPE_CS SIMONE_OBJTYPE_CV SIMONE_OBJTYPE_PIPE SIMONE_OBJTYPE_RECP SIMONE_OBJTYPE_VA SIMONE_OBJTYPE_SE SIMONE_OBJTYPE_RE SIMONE_OBJTYPE_NS SIMONE_OBJTYPE_NO SIMONE_OBJTYPE_MACRO SIMONE_OBJTYPE_UNIT SIMONE_OBJTYPE_SYS SIMONE_OBJTYPE_MS SIMONE_OBJTYPE_NRV
SIMONE_FLAG_FILTER_PARAM_TYPE	SIMONE_PARAM_TYPE_SETPOINTS SIMONE_PARAM_TYPE_SUPPLY_OFFTAKE (Q) SIMONE_PARAM_TYPE_QUALITY SIMONE_PARAM_TYPE_MEASUREMENTS (PM, PMDP, MM, SIGMA)

Return values:

simone_status_nolicense	no license
simone_status_badpar	parameter invalid
simone_status_ok	ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

9.9 simone_get_entry_reset_filter()

This function resets all filters for the following calls of the *simone_get_entry()* function previously set with the *simone_get_entry_set_filter()* function

Return values:

simone_status_nolicense	no license
simone_status_ok	ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

9.10 simone_get_entry(rtime, obj_id, ext_id, cond_flags, cond_id, value, value_str, value_str_len, unit, func_id, value_flags, src_id, comment, comment_len)

Unlike the other read functions, this interface does not read the results data, but reads back the scenario entries, i.e. the input parameters only. It requires the scenario to be open in READ mode. The entries are read sequentially in time, starting from the retrieval time set. As after opening a scenario, this is at the end time, it must be set to zero with the *simone_set_rtime* function if all entries from the scenario shall be read. If the start time is set to another value, all entries before this time are skipped.
This interface delivers all items of an entry including conditions (set by *cond_id* and *cond_flags*), references to a function or load profile (set by *func_id*) to define a value, as well as added comment text or marked entries belonging to a 'source' (set by *src_id*).

A condition is expressed by a function (referenced here by *cond_id*) , that will be evaluated at run-time, and a flag (set as *cond_flags*) that defines how the result shall be used to decide whether or not the scenario parameter should take effect. The flag values are defined here by symbolic names that are similar to how the values are shown at the user interface.

Note:
Identifiers of load profiles and source-identifiers are always 'local' and only valid for the actual scenario. Load profiles as described in the chapter Load Profiles and Functions are defined by special entries that are not returned by simone_get_entry. Rather, use *simone_begin_read_profile* and *simone_read_profile* to retrieve load profiles. Functions may be 'local' or 'global' depending on how they were defined. Use *simone_get_function* to determine this attribute of a function. *simone_id2name* is provided to determine the type of an object, in particular to decide whether it is a profile or function. If the information delivered for an entry is to be used to create a new scenario, be sure to translate the local objects to their text representation first and create new identifiers from this for the destination scenario. Only these new local identifiers can be supplied to *simone_write_ex* for a new scenario. (see the section Load Profiles and Functions and *simone_define_source_name*).

The parameter *value* is returned depending on the type of the entry: All values of type float are returned by the *value* parameter and the *value_str* parameter is filled with an empty string. All others are returned by the *value_str* parameter like with *simone_read_str*. For entries which do not take any value (e. g.: <Control valve>.OFF or values defined with a function evaluated at run-time), the *simone_get_entry* function returns simone_status_nofloat and the *value_str* parameter is filled with an empty string.

If the delivered *func_id* set to the id of a load profile, the scenario parameter addressed by *obj_id, ext_id* is a boundary flow. In this case, the flow is defined as the value of the load profile at the actual time multiplied by the factor delivered here by *value.* An entry created this way delivers no *rtime* (*rtime is set to 0),* unless it references the special profile *const*.

The profile *const* is pre-defined and represents a constant 1, i.e. the flow is defined by the value at the time delivered here in *rtime*. This way to define the time-series of boundary flow is used if data are transferred from e.g. a SCADA system at regular intervals. Use simone_varid("CONST.DEF",…) to get the object id for this profile when writing such an entry by *simone_write_ex*.

With the *simone_get_entry_set_filter()* function the entries to be retrieved can be filtered to return only dedicatedl entries from the scenario.

Parameters:

rtime	- the parameter is valid from this time if rtime is 0, parameter is valid from the start for entries in static scenarios the rtime parameter always returns 0	OUT

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

obj_id	- object-ID of parameter		OUT
ext_id	- extension-ID of parameter		OUT
cond_flags	- SIMONE_NO_FLAG:	used if no cond_id is set	OUT
	SIMONE_FLAG_IF:	parameter is set if condition is valid	
	SIMONE_FLAG_IF1:	parameter is set if condition is first time valid	
	SIMONE_FLAG_IFN:	parameter is set if condition is NOT valid	
	SIMONE_FLAG_IF1N:	parameter is set if condition is first time NOT valid	
cond_id	- id of function to be combined with cond_flag as condition 0: no condition used		OUT
value	- float value of parameter		OUT
value_str	- value as formatted character string, if available and not a float else an empty strmg: refer to *simone_write_attr, simone_write_configuration* for special formats		OUT
value_str_len	- maximum length for value string including terminating zero		IN
unit	- unit descriptor for value		OUT
func_id	- id function or profile to be combined with the value 0: no function nor profile used		OUT
value_flags	- SIMONE_NO_FLAG		OUT
	SIMONE_FLAG_INVALID	- value marked as invalid	
src_id	- id of local name used as source name source names are used to build groups of parameters within a scenario 0: no source name used		OUT
comment	- comment for parameter empty string or NULL: no comment used		OUT
comment _len	- maximum length for comment including terminating zero		IN

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

Return values:

simone_status_nolicense	no license
simone_status_nofile	no scenario open
simone_status_badseq	no network selected scenario only open for reading
simone_status_invid	object-ID and extension-ID do not fit
simone_status_badpar	any other parameter invalid
simone_status_locked	attributes are already in use by another application
simone_status_not_found	ok, no more entries in scenario
simone_status_ok	ok, float value delivered
simone_status_nofloat	ok, no float value delivered

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

9.11 simone_position_info(obj_id, distance_x, unit, obj_id_1, obj_id_2, distance_z, distance_1_2, flags,)

This special function determines the position of a specified location in the network relative to nearby non-pipe elements, i.e. normally the closest valves.
The location is specified either as a node or a position on a pipe element (like e.g. the leak location indicates the location of a leak).
The position returned is indicated as a distance_z from the start of the shortest path between two nearby objects, normally valves or other non-pipe elements, which are also determined and returned by this function. If no non-pipe element can be reached to either side of the location, the next terminal node is returned instead.

Note:
If the object specified with obj_id is a pipe, the distance_x is to be specified relative to the start node of the pipe. To find out, which node is the start of a pipe, use the function dd_varid(name, obj_id, ext_id), where name is <name of pipe>.I.<extension>.

Parameters:

obj_id	- object id of specified node or pipe	IN
distance_x	- distance relative to start of specified pipe, only interpreted if object is a pipe	IN
Unit	- unit descriptor for all distance values	IN
obj_id_ 1	- first terminating element of path or terminal node	OUT
obj_id_ 2	- second terminating element of path or terminal node if no element or node can be found, obj_id_2 = 0 is returned	OUT
distance_z	- distance of indicated location from obj_id_1	OUT
distance_1_2	- distance between obj_id_1 and obj_id_2 invalid, if returned obj_id_2 == 0	OUT
Flags	- reserved for future use	IN

Return values:

simone_status_nolicense	no license
simone_status_badseq	no network selected
simone_status_badpar	wrong object type of obj_id_leak_node
simone_status_not_found	no path found including margins and node next to leak
simone_status_ok	Ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

9.12 simone_get_quality_in_path(path_id, ext_id, unit_quality, value, unit_position, position, max_buf, filled_buf, flags)

Read positions and values of quality flags along a specified path, route or pipe for the actual retrieval time. The positions are given as distance from the start node of the path or pipe.

Parameters:

path_id	- id of object set forming a path or route or id of pipe	IN
ext_id	- id of a quality parameter to be retrieved	IN
unit_quality	- unit descriptor for quality parameter values to be retrieved	IN
value	- array of quality flag values	OUT
unit_position	- unit descriptor for quality flag positions to be retrieved (of unit type SIMONE_UNIT_TYPE_L)	IN
position	- array of quality flag positions relative to start of path, route or pipe	OUT
max_buf	- maximum number of values and positions to be returned in the respective arrays	IN
filled_buf	- returned number of flags	OUT
flags	- reserved for future use	IN

Return values:

simone_status_badseq	SIMONE API is not initialized, network not selected
simone_status_nolicence	no or wrong license
simone_status_nofile	no scenario open
simone_status_badpar	arrays for results too small
simone_status_not_found	no quality flags in object set with path or route
simone_status_ok	Ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

9.13 simone_get_quality_in_pipe_array(n_pipes, pipe_ids, ext_id, unit_quality, value, unit_position, position, max_buf, filled_buf, offset_buf, stats, flags)

Read positions and values of quality flags for a number of pipes

For each pipe a number of values and a number of positions relative to the start of the pipe are retrieved. These are concatenated in the returned arrays and the offset_buf contains the offsets to the start of the list of values for each pipe.

Parameters:

n_pipes	- count of values for the following arrays: pipe_ids, offset_buf, stats	IN
pipe_ids	- array with IDs of pipes	IN
ext_id	- id of a quality parameter to be retrieved	IN
unit_quality	- unit descriptor for quality parameter values to be retrieved	IN
value	- concatenated array of quality flag values for all pipes	OUT
unit_position	- unit descriptor for quality flag positions to be retrieved (of unit type SIMONE_UNIT_TYPE_L)	IN
position	- concatenated array of quality flag positions relative to start of each pipe	OUT
max_buf	- maximum number of values and positions to be returned in the respective arrays	IN
filled_buf	- returned number of flags	OUT
offset_buf	- offsets to first position, offsets to first start value for each pipe	OUT
stats	- array of status indicators for each pipe information	OUT
flags	- reserved for future use	IN

Return values:

simone_status_badseq	SIMONE API is not initialized, network not selected
simone_status_nolicence	no or wrong license
simone_status_nofile	no scenario open
simone_status_badpar	arrays for results too small
simone_status_not_found	no quality flags in pipe
simone_status_ok	Ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

10 Writing scenario entries

Writing applies to the current scenario, which must have been opened in CREATE or WRITE mode.

A scenario is defined at the user interface as a table of scenario parameter settings. The API provides interfaces to write such entries for the current scenario.

Mainly the entries of this table assign float values to scenario parameters. Such entries can be created by the *simone_write…* functions described in this section.

Some parameters (denoting the control mode or state of an element) do not take values, but rather their extension (e.g. BP, ON; OFF) already defines the setting. Also for these entries *simone_write…* can be used, as the value simply can be left undefined.

As the ON and OFF states are also reflected as values of the variable <element>.MODE upon reading, for symmetry and convenience the *simone_write…* interfaces support writing this variable using the special values SIMONE_CTRL_ON and SIMONE_CTRL_OFF. This will result in generating an entry <valve>.ON or <element>.BP (for CS/CV elements) or <element>.OFF accordingly.

A further extension of this concept is provided for ease of online interface programming. If for <element>.MODE another control mode value like e.g. SIMONE_CTRL_SPO is written, this does not result in a scenario entry, but subsequent writing of setpoints other than matching the defined mode are ignored. This allows processing the full list of assigned setpoint data from a 'request list', while only the currently valid - indicated by the mode - is passed to the scenario definition.
By setting the flag SIMONE_FLAG_FORCE_OVERWRITE in the function simone_open() before or in the simone_write…() functions, writing mode values for ON, OFF, BP and writing setpoint values can be enforced against the above rules. If SIMONE_FLAG_FORCE_OVERWRITE flag is set, the last written value becomes valid.
.
The values of other parameters (like QT, LAMBDA, etc.) are handled at the user interface by discrete text strings ("ON", "HOFER", …), but internally are dealt with by float values as well.
To support creating entries for these by *simone_write* as well, symbolic names for the internal values are provided in simone_api.h, simone_api.bas, simone_api.net respectively.

More sophisticated entries in a scenario may refer functions or load profiles. To allow creating such entries, *simone_write_ex* is available. The functions and load profiles can be manipulated by the interfaces described in the section 'Functions and Load Profiles'. These interfaces also yield the necessary id's, which are used to reference these items in creating a scenario entry.

The data_type of a parameter can be read with the *simone_varid_info* function.
Depending on the data_type the following *simone_write* functions are used:

Parameters with data_type is SIMONE_DATA_FLOAT can be set with one of the *simone_write*,*simone_write_with_flag, simone_write_array()* or *simone_write_ex* functions.

Parameters defining a configuration of a compressor station can be set either with the *simone_write_configuration* or with the *simone_write_ex* function using the value_str parameter.

For user-defined attributes all data_types are possible. User defined attributes are set either with the *simone_write_attr* or *simone_write_ex* function using the value_str parameter. The format of the value_str parameter is described with the *simone_write_attr* function.

The flag SIMONE_FLAG_ATTR2SCENARIO enforces to rwrite attribute values of numeric type as scenario parameters (like in the GUI). This effectively overwrites an attribute value locally for a scenario. If this flag is not, attribute values are global and saved only once for the whole network (for all scenarios).

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

The flag may be set individually for *simone_write_with_flag(), simone_write_ex()* and *simone_write_array(),* or,if set in simone_open(), it takes effect for all subsequent writes to the scenario.

The time for a scenario entry (i.e. since when the parameter should take effect for the simulation) needs to be specified in each call. The time must be after the start of the scenario or 0 (= no time), if the parameter should be valid since the very beginning.

When defining entries for an object no contradictory parameters are allowed, e. g. a controlled valve cannot be controlled by a flow setpoint and pressure setpoint at the same time. Hence, if a respective entry for the same time already exists, the write interfaces by default simply replace it with the entry defined in the call. However, special rules apply for entries defining the 'mode' of an element – like ON/OFF/BP or SPO, SM, etc. respectively. As outlined above, for these it is possible to first write a <element>.mode (e.g. with a value of SIMONE_CTRL_SPO) and then other entries with different setpoints. Due to the first call, the write interface will ignore all subsequent writings of setpoints other than that single one (<element>.SPO in this example) matching the defined mode.

A unit descriptor (or SIMONE_UNIT_DEFAULT) must be supplied to indicate in which unit the supplied value is to be interpreted. The unit_type of a parameter can be read with the *simone_varid_info* function, use *simone_unit2des* to generate a valid unit descriptor. If SIMONE_UNIT_DEFAULT is supplied, the value will be interpreted in the unit set by *simone_set_api_default_unit* or in the default unit that was set at the time of *simone_init* from the configuration.
For upward compatibility with earlier versions, also the symbolic names for units of flow or pressure as defined in simone_api.h, simone_api.bas, simone_api.net can be supplied instead of a unit descriptor.

If no unit descriptor is necessary for a value to be written, use SIMONE_UNIT_DEFAULT.

If real time data are to be supplied for reconstruction or leak detection, it may happen that data are temporarily unavailable or bad. To indicate this to SIMONE, a scenario parameter can be written with specifying the flag SIMONE_FLAG_VALUE_INVALID.
Further flags supported (with *simone_write_with_flag()* or *simone_write_ex(), simone_array()*) are SIMONE_FLAG_VALUE_INACTIVE to create a deactivated entry like at the user interface or SIMONE_FLAG_VALUE_CHECK to instruct the interface to (range-)check the value.

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

10.1 simone_write(rtime, obj_id, ext_id, value, unit)

Write a simple scenario parameter.

For parameters, which do not need a value (e. g.: <valve>.OFF), use value = 0.0 and unit = SIMONE_UNIT_DEFAULT.

For simple control modes (ON,OFF,BP) also writing <object>.MODE with the special values SIMONE_CTRL_ON and SIMONE_CTRL_OFF is supported. This will result in generating an entry like <object>.ON/OFF/BP respectively. Writing other control modes may be also meaningful as described above at the beginning of this chapter.
If writing boundary flows (.Q) for nodes, *simone_write* does a special handling for the convenience of SCADA interface applications The value then is written by referencing the special profile *const*, which is pre-defined and represents a constant 1. This way to define a boundary flow is used if data are transferred from e.g. a SCADA system at regular intervals. The times for which data are written this way must match the cycle time(s) defined in the configuration prepared by the online environment – see the data_exchange documentation for more detail.
To write a standard setting for Q or referencing a real profile, use *simone_write_ex.*

Parameters:

rtime	- time for the parameter is valid from this time if rtime is set to 0, parameter is valid from the start for static scenarios the rtime parameter is ignored	IN
obj_id	- object-ID of parameter	IN
ext_id	- extension-ID of parameter	IN
value	- value to be set for parameter	IN
unit	- unit descriptor for value or SIMONE_UNIT_DEFAULT	IN

Return values:

simone_status_nolicense	no license
simone_status_nofile	no scenario open
simone_status_badseq	no network selected scenario only open for reading attribute values cannot be written, if an edited version of the network exists
simone_status_invid	object-ID and extension-ID do not fit
simone_status_badpar	invalid value or unit
simone_status_ok	ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

10.2 simone_write_with_flag(rtime, obj_id, ext_id, value, unit, flag)

Write a scenario parameter with flags. SIMONE_FLAG_VALUE_INVALID creates a scenario entry that is meaningful only for metered values (.Q, .PM or .MM). It is used to signal to SIMONE reconstruction or leak detection, that a metering station at a node exists, but the metered value is currently bad.
Use SIMONE_FLAG_VALUE_INACTIVE to create a deactivated entry like at the user interface or SIMONE_FLAG_VALUE_CHECK to instruct the interface to (range-)check the value supplied.

If SIMONE_NO_FLAG is set, this interface works identical to *simone_write*.

Parameters:

Rtime	- the parameter is valid from this time if rtime is set to 0, parameter is always valid for static scenarios the rtime parameter is ignored	IN
obj_id	- object-ID of parameter	IN
ext_id	- extension-ID of parameter	IN
Value	- value to be set for parameter	IN
Unit	- unit descriptor for value or SIMONE_UNIT_DEFAULT	IN
Flag	- SIMONE_NO_FLAG - SIMONE_FLAG_VALUE_INVALID – mark value as invalid - SIMONE_FLAG_VALUE_INACTIVE - entry is inactive ('commented') - SIMONE_FLAG_VALUE_CHECK – check value if in valid range - SIMONE_FLAG_FORCE_OVERWRITE – enforce writing mode value or setpoint - SIMONE_FLAG_ATTR2SCENARIO – write attribute as scenario parameter	IN

Return values:

simone_status_nolicense	no license
simone_status_nofile	no scenario open
simone_status_badseq	no network selected or scenario only open for reading attribute values cannot be written, if an edited version of the network exists
simone_status_invid	object-ID and extension-ID do not fit
simone_status_badpar	invalid value or unit
simone_status_noval	invalid value (out of range)
simone_status_ok	Ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

10.3 simone_write_array(rtime, n_params, obj_ids, ext_ids, values, units, flags, stats)

This interface provides the same function as *simone_write_with_flag*, but for an array of SIMONE parameters, thus reducing the overhead that may become relevant if a larger number of values is to be written, in particular when working with the remote API.
The values of *n_params* parameters as specified by obj_ids, ext_ids are written for the specified rtime. For each parameter in the list, also a unit descriptor and a flag is to be specified.
If all parameters are successfully written, the function returns with the ok status. If the license is not sufficient, no scenario is open or no network is selected, nothing can be written and an appropriate status is returned. In case there are problems with one or more elements of the list, a respective status is returned, and the stats array can be checked to identify which list elements have a problem.

Parameters:

rtime	- the parameter is valid from this time if rtime is set to 0, parameter is always valid for static scenarios the rtime parameter is ignored	IN
n_params	- number of parameters to be set	IN
obj_ids	- array of object-IDs of parameters	IN
ext_ids	- array of extension-IDs of parameters	IN
values	- array of values to be set for parameters	IN
units	- array of unit descriptors for values or SIMONE_UNIT_DEFAULT	IN
flags	- array of flags to be set for parameters SIMONE_NO_FLAG SIMONE_FLAG_VALUE_INVALID mark value as invalid SIMONE_FLAG_VALUE_INACTIVE entry is inactive ('commented') SIMONE_FLAG_VALUE_CHECK check value if in valid range SIMONE_FLAG_FORCE_OVERWRITE enforce writing mode value or setpoint value SIMONE_FLAG_ATTR2SCENARIO – write attribute as scenario parameter	IN
stats	- array of status indicators for each parameter	OUT

simone_status_invid	object-ID or extension-ID do not fit
simone_status_badpar	invalid value or unit
simone_status_noval	invalid value (out of range)
simone_status_ok	Ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

Return values:

simone_status_nolicense	no license, no parameter written
simone_status_nofile	no scenario open, no parameter written
simone_status_badseq	no network selected or scenario only open for reading, no parameter written attribute values cannot be written, if an edited version of the network exists
simone_status_invid	at least one object-ID or extension-ID do not fit
simone_status_badpar	at least one invalid value or unit
simone_status_noval	at least one invalid value (out of range)
simone_status_ok	Ok

10.4 simone_write_configuration(rtime, obj_id, ext_id, configuration)

Write a scenario parameter for the configuration of a compressor station. The configuration is to be supplied as a text string in the same format as it would appear at the user interface.

Parameters:

rtime	- the parameter is valid from this time if rtime is set to 0, parameter is valid from the start for static scenarios the rtime parameter is ignored	IN
obj_id	- object-ID of compressor station	IN
ext_id	- extension-ID	IN
configuration	- string value, e. g.: FREE, GENERIC, ‘12’ refer to user manual for further information	IN

Return values:

simone_status_nolicense	no license
simone_status_nofile	scenario not open
simone_status_badseq	no network selected scenario only open for reading
simone_status_badpar	object-ID and extension-ID do not fit invalid configuration configuration could not be written
simone_status_ok	ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

10.5 simone_write_ex(rtime, obj_id, ext_id, cond_flags, cond_id, value, value_str, unit, func_id, value_flags, src_id, comment)

Write a scenario entry to the current scenario. The basic functionality is as described for *simone_write*. But this interface additionally allows to write conditioned entries (by setting *cond_id* and *cond_flags*), to define a value by referencing a function or load profile (by setting *func_id*), as well as to add a comment text or to mark the entry as belonging to a 'source' (by setting *src_id*).

All referred items (function, profile, src_name) must be created before, either at the user interface or with the respective API functions (see the section Functions and Load Profiles, section Objects, Names and IDs for src_names), and be referenced here by their id.

A condition is expressed by a function (referenced here by *cond_id*) , that will be evaluated at run-time, and a flag (set as *cond_flags*) that defines how the result shall be used to decide whether or not the scenario parameter should take effect. The flag values are defined here by symbolic names that are similar to how the values are shown at the user interface.

For the parameter value two formats are supported: For all values in SIMONE which are of float type, the (float) value parameter can be used, but it is also possible to use the value_str parameter, which is a string type. So also configurations of compressor stations and user-defined attributes are handled with this function. (For more details refer to the description of the *simone_write_configuration* and *simone_write_attr* function.)

If *func_id* is set to an id of a function, the value of the function as evaluated at run-time will be used for the scenario parameter. In this case, the parameter *value* is ignored.

If *func_id* is set to the id of a load profile, the scenario parameter addressed by *obj_id, ext_id* must be a boundary flow. In this case, the flow will be set as the value of the load profile at the actual time multiplied by the factor set here by *value.* An entry created this way should not define a time (i.e. set *rtime=0),* unless it references the special profile *const*.

The profile *const* is pre-defined and represents a constant 1, i.e. the flow is defined by the value at the time specified here in *rtime*. This way to define the time-series of boundary flow is used if data are transferred from e.g. a SCADA system at regular intervals. The times for which data are written this way must match the cycle time(s) defined in the configuration prepared by the online environment – see the data_exchange documentation for more detail.

When defining entries for an object no contradictory parameters are allowed, e. g.: a controlled valve cannot be controlled by a flow setpoint and pressure setpoint at the same time. Hence, if a respective entry for the same time already exists, *simone_write_ex* may replace it with the entry defined in the call, subject to the rules outlined at the beginning of this chapter.
For simone_write_ex, the decision of what entries may are in conflict need to be extended by considering the *func_id* and *condtion* depending on the type of entry:

1. mode or setpoint entries (e. g. <valve>.ON or <cv>.SPO or <cs>.SM) are regarded as conflicting and subject to replacement rules if they are equal in rtime, obj_id, condition
2. boundary condition parameter entries (supply or offtake quantities) are replaced, if they are equal in rtime, obj_id, ext_id, condition, func_id
3. all other entries are replaced, if they are equal in rtime, obj_id, ext_id, condition

Parameters:

Rtime	- the parameter is valid from this time if rtime is set to 0, the parameter is valid from the start	IN

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

	for static scenarios the rtime parameter is ignored	
obj_id	- object-ID of parameter	IN
ext_id	- extension-ID of parameter	IN
cond_flags	- SIMONE_NO_FLAG: use if no cond_id is set SIMONE_FLAG_IF: parameter is set if condition is valid SIMONE_FLAG_IF1: parameter is set if condition is first time valid SIMONE_FLAG_IFN: parameter is set if condition is NOT valid SIMONE_FLAG_IF1N: parameter is set if condition is first time NOT valid	IN
cond_id	- id of function to be combined with cond_flag as condition 0: no condition used	IN
Value	- value for parameter to be set if no value necessary: set to 0.0 if value_str parameter is filled, the value parameter is not interpreted	IN
value_str	- value in string format for parameter to be set refer to *simone_write_attr, simone_write_configuration* for special formats if not empty: value parameter is not interpreted NULL or empty string: value parameter is interpreted	IN
Unit	- unit descriptor for value or SIMONE_UNIT_DEFAULT	IN
func_id	- id function or profile to be combined with the value profile is only valid for boundary condition parameters (supply- and offtake quantities). 0: no function nor profile used	IN
value_flags	- SIMONE_NO_FLAG - SIMONE_FLAG_VALUE_INVALID - mark value as invalid - SIMONE_FLAG_VALUE_INACTIVE - entry is inactive ('commented') - SIMONE_FLAG_VALUE_CHECK - check if value in valid range - SIMONE_FLAG_FORCE_OVERWRITE - enforce writing mode value or setpoint value - SIMONE_FLAG_ATTR2SCENARIO - write attribute as scenario parameter	IN
src_id	- id of local name used as source name source names are used to build groups of parameters within a scenario 0: no source name used	IN
comment	- comment for parameter empty string or NULL: no comment used	IN

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

Return values:

simone_status_nolicense	no license
simone_status_nofile	no scenario open
simone_status_badseq	no network selected scenario only open for reading attribute values cannot be written, if an edited version of the network exists
simone_status_invid	object-ID and extension-ID do not fit
simone_status_badpar	any other parameter invalid
simone_status_noval	invalid value (out of range)
simone_status_locked	attributes are already in use by another application
simone_status_ok	ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

11 Merge and Erase Scenario Entries

Merging the entries of another scenario into the current one is possible at the user interface and also supported in the API using the function *simone_include_scenario*.

Profiles or groups of scenario entries with the same source identifier can be deleted using *simone_erase_entries*.

Including scenario entries offers to repeatedly re-use the same set of parameters for different scenarios. To mark the included entries, by default the name of the source scenario is used. When re-including the same source a second time, first all entries in the destination marked with this source name are removed, unless the flag SIMONE _FLAG_NO_REPLACE is set.
If this flag is set, only parameters not yet defined in the destination (for the same time) are added.

A special handling is provided when setting SIMONE_FLAG_CLEAR_BEFORE_INSERT.
Then first all entries in the current scenario are removed, that are defining the same parameter for the same object as any entry in the source scenario. This allows to be sure that the source definitions will exclusively set up the respective parameters for those objects.
This removal procedure will be effective even if the SIMONE_FLAG_NO_REPLACE flag is set.

Including entries can be restricted to a time interval (*start_time, end_time*) related to the source scenario, if the source scenario is of type dynamic.

If the current (destination) scenario is of a static type, only the last scenario entry valid before *end_time* is included for a parameter into the current scenario.

If the source scenario is of a static type and is to be included into a current of a dynamic type, all entries are included for *start_time*.

When both (source and destination) are of a dynamic type, the times of scenario entries are by default not changed when copied. As SIMONE internally uses hours since midnight from the starting date of the respective scenario, the copied entries are then valid as for the starting date of the current scenario. The flag SIMONE_FLAG_SHIFT_TIME_RESPECT_DATE can be set to have different starting dates of both scenarios respected (i.e. the entries copied are inserted according to their date and time as of the source scenario).

The flag SIMONE_FLAG_SET_NOTIME_BEFORE_INITIME can be set to force setting the time to *notime* (= valid from the start time of the current scenario) for entries with a time before the *initime* of the current scenario.

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

11.1 simone_include_scenario(src_network, src_scenario, start_time, end_time, src_id, flags)

Include entries from another scenario (source) to the current scenario (destination).
The current scenario must be open in CREATE or WRITE mode. The source scenario may also be located in another network.

The rules for normal scenario entries are outlined above. Also functions and profiles are included, as far as they are not yet defined in the current scenario.

If a profile already exists in the current scenario, it is overwritten, i.e. all profile entries in the current scenario are replaced with the profile entries included from the source scenario, unless the flag SIMONE _FLAG_NO_REPLACE is set. The time interval takes no effect on profiles. Also in no case a time adjustment occurs.

If a (local) function is already defined in the current scenario, the function definition of the current scenario will remain. The overwrite flag takes no effect for function definitions.

Because names for local functions, load profiles, macros, and source names must be unique, it is not possible to include a function or profile, if its name is already in use for another type in the current scenario.

Entries marked as inactive in the source are not included.

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

Parameters:

src_network	- name of source network with scenario to be included NULL or empty string: use current network	IN
src_scenario	- name of source scenario	IN
start_time	- start of time interval for entries to be included or 0: no restriction	IN
end_time	- end of time interval for entries to be included or 0: no restriction	IN
src_id	- id of source name to mark all included records 0: mark included records with name of source scenario	IN
flags	- SIMONE_NO_FLAG (default) SIMONE_FLAG_SHIFT_TIME_RESPECT_DATE Entries copied are included according to their date + time SIMONE_FLAG_SET_NOTIME_BEFORE_INITIME set time stamp to 0 (notime, valid from the beginning) for all included entries with time stamp before initime of current scenario SIMONE_FLAG_CLEAR_BEFORE_INSERT first remove all entries in actual scenario, where object and parameter match any such entry in the source SIMONE _FLAG_NO_REPLACE do not overwrite entries in current scenario, except those, which are removed due to SIMONE_FLAG_CLEAR_BEFORE_INSERT being set	IN

Return values:

simone_status_nolicense	no license
simone_status_nofile	no scenario open
simone_status_badseq	no network selected scenario only open for reading
simone_status_badpar	invalid src_id, name already in use for local function, load profile, macro invalid src_network invalid src_scenario
simone_status_error	internal error
simone_status_ok	ok

11.2 simone_scenario_save_as(new_scenario, scenario_type, inic_file, comment, flags)

Save a copy of the actual scenario under another name as a new scenario and open new scenario for writing as actual scenario.

As at the user interface, at this occasion the type of the new scenario can be changed. If this changes from a static to a dynamic type, the time stamp of all entries is set to the start_time of the scenario. If the type changes from dynamic to static, all parameter values as valid at end_time in the actual scenario are saved in the new scenario.

Parameters:

new_scenario	- name of new scenario to be created	IN
scenario_type	- type of new scenario to be created SIMONE_RUNTYPE_DYN dynamic simulation SIMONE_RUNTYPE_REC reconstruction SIMONE_RUNTYPE_STA static simulation SIMONE_RUNTYPE_FIL filter SIMONE_RUNTYPE_S_O set point optimization SIMONE_RUNTYPE_C_O configuration optimization SIMONE_RUNTYPE_PER periodic run or 0: set type from actual scenario	IN
inic_file	- initial state of new scenario to be created empty string or NULL: set initial state from actual scenario or set INIT for set point optimization, configuration optimization	IN
comment	- comment of new scenario to be created empty string or NULL: set comment from actual scenario	IN
flags	- SIMONE_FLAG_VISIBLE (default) SIMONE_FLAG_INVISIBLE	IN

Return values:

simone_status_nolicense	no license
simone_status_nofile	no scenario open
simone_status_badseq	no network selected new scenario already existing
simone_status_badpar	wrong parameter new_scenario wrong parameter inic_file invalid type of new scenario
simone_status_ok	ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

11.3 simone_erase_entries(start_time , end_time, filter_id, filter_flag)

Delete entries from the actual scenario. .

The erasing of entries can be restricted to a particular profile or to entries marked by a source name as indicated by a respective id in *filter_id*.

Erasing can additionally be restricted to the time interval given by *start_time, end_time*.

If the filter is set to a profile, the profile definition as well as all entries referring the profile are being erased, unless the flag SIMONE_FILTER_ONLY_REFERENCES is set. This flag will restrict erasing to entries referring the profile, and NOT to erase the profile definition itself.

If no filter is set, all entries in the given time interval are deleted and the filter flag SIMONE_FILTER_ONLY_REFERENCES takes no effect.

Parameters:

start_time	- start of time interval for entries to be erased or 0: no restriction	IN
end_time	- end of time interval for entries to be erased or 0: no restriction	IN
filter_id	- id of profile or source name to be filtered 0: no filter is used, all entries in selected time interval are erased	IN
filter_flag	- SIMONE_NO_FLAG (default) profile is removed and all entries which refer profile or source name are removed SIMONE_FILTER_ONLY_REFERENCES only entries which refer profile or source name are removed, profile definitions are left untouched SIMONE_ERASE_LOAD_PROFILES remove all load profiles, if no filter_id set	IN

Return values:

simone_status_nolicense	no license
simone_status_nofile	no scenario open
simone_status_badseq	no network selected scenario only open for reading entry cannot be erased because other entries refer to it wrong type of open scenario
simone_status_badpar	invalid filter_id
simone_status_ok	ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

11.4 simone_erase_entry(rtime, obj_id, ext_id, cond_id, flags)

Delete a specified entry and/or specified conditioned entries from the actual scenario

A normal (unconditioned) entry to be erased must be specified with time, object ID and extension ID, setting cond_id=0 and specifying SIMONE_NO_FLAG.

If a conditioned entry shall be deleted, additionally the cond_id needs to be specified. The flags parameter then has to be set to the respective condition flag(s).
If SIMONE_FLAG_ERASE_CONDITIONS is specified, all matching entries are erased (uncoditioned entry and all conditioned).

Parameters:

rtime	- time for entries to be erased if rtime is set to 0, parameter is valid from the start of the scenario for static scenarios the rtime parameter is ignored		IN
obj_id	- object ID for entries to be erased		IN
ext_id	- extension ID for entries to be erased special handling for SIMONE_EXT_MODE: erase entries for any mode (ON, OFF, BP, SPO, SPI, ...)		IN
cond_id	- condition ID for entries to be erased		IN
flags	- if no condition specified (con_id = 0)		IN
	SIMONE_NO_FLAG	(default)	
	SIMONE_FLAG_ERASE_CONDITIONS		
	if a condition is specified (cond_id ≠ 0)		
	SIMONE_FLAG_IF	IF VALID	
	SIMONE_FLAG_IF1	FIRST IF VALID	
	SIMONE_FLAG_IFN	IF NOT VALID	
	SIMONE_FLAG_IF1N	FIRST IF NOT VALID	
	SIMONE_FLAG_IF_ALL	(default)	

Return values:

simone_status_nolicense	no license
simone_status_nofile	no scenario open
simone_status_badseq	no network selected scenario scenario only open for reading
simone_status_not_found	no matching entry found
simone_status_ok	ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

12 User-defined attributes

User-defined attributes can be defined per network either at the user interface or using the respective API interfaces described below (simone_create_attr_def, …)..

Reading attributes may be accomplished using the standard read functions *simone_read, simone_read_str.* For writing user defined attributes either the *simone_write_ex* can be used and a special routine *simone_write_attr* is provided, because attributes may have more distinct data types. However, attributes of numeric type (int or float) can also be written as scenario parameters (like in the GUI).when setting the flag SIMONE_FLAG_ATTR2SCENARIO in the call to simone_open or a call for writing a single value.This effectively overwrites an attribute value locally for a scenario. If this flag is not set, attribute values remain global and are saved only once for the whole network (for all scenarios).

Writing attributes by *simone_write_attr* does not require a scenario to be open - attributes apply to the network object only and cannot have different values at different times or for different scenarios.

Except for *simone_read*, attribute values, including the defaults and limits are dealt with by their text representation and the contents of a value string at the interface must match the defined attribute type:

attribute type	Content of string
SIMONE_DATA_INT	Integer Number
SIMONE_DATA_FLOAT	Floating Number with Decimal Point
SIMONE_DATA_STRING	Characers
SIMONE_DATA_BOOL	"True" | "False" | “1” | “0” | “1.0” | “0.0”
SIMONE_DATA_DATETIME	"<day>.<month>.<year> <hour>:<minute>" if year < 1970, add 100 (1923 ➔ 2023, 30 ➔2030) 1.1.1970 0:00 (UTC) <= allowed date

The length of all textual parameters (attr_name, attr_ext_name, default_value and limits) is limited to the same length as with the user interface.

Note:
Unlike the user interface the SIMONE API functions allow to handle attributes which are used internally by SIMONE and hidden at the user interface. Such hidden attributes should be treated with care and preferably not being touched. All attributes marked with the attr_flag SIMONE_FLAG_VISIBLE can be changed by the user.

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

12.1 simone_create_attr_def(attr_name, attr_ext_name, attr_flags, attr_mask, attr_type, default_value, lower_limit, upper_limit, ext_id, flags)

Create a new attribute definition in the attribute dictionary.

Parameters:

attr_name	- description for attribute definition to be created		IN
attr_ext_name	- extension for attribute definition to be created		IN
attr_flags	- SIMONE_NO_FLAG	attribute hidden for SIMONE GUI, attribute for internal SIMONE usage do not modify	IN
	. SIMONE_FLAG_VISIBLE	Visible and can be modified by user	
attr_mask	- SIMONE_OBJTYPE_CS SIMONE_OBJTYPE_CV SIMONE_OBJTYPE_PIPE SIMONE_OBJTYPE_RECP SIMONE_OBJTYPE_VA SIMONE_OBJTYPE_SE SIMONE_OBJTYPE_RE SIMONE_OBJTYPE_NS SIMONE_OBJTYPE_NO ..SIMONE_OBJTYPE_MS SIMONE_OBJTYPE_NRV SIMONE_OBJTYPE_SUB SIMONE_OBJTYPE_UNIT		IN
attr_type	- SIMONE_DATA_INT SIMONE_DATA_FLOAT SIMONE_DATA_STRING SIMONE_DATA_BOOL SIMONE_DATA_DATETIME		IN
default_value	- default_value for attribute definition to be created empty string or NULL: do not define a default for values		IN
lower_limit	- lower_limit for attribute definition to be created empty string or NULL: do not define a lower limit for values		IN
upper_limit	- upper_limit for attribute definition to be created empty string or NULL: do not define an upper limit for values		IN
ext_id	- Extension ID of attribute definition created		OUT
flags	- reserved for future use		IN

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

Return values:

simone_status_nolicense	no license
simone_status_badseq	no network selected attribute values cannot be written, if an edited version of the network exists
simone_status_badpar	invalid parameter
simone_status_locked	attributes are already in use by another application
simone_status_ok	ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

12.2 simone_modify_attr_def(ext_id, attr_name, attr_flags, attr_mask, default_value, lower_limit, upper_limit, flags)

Modify an attribute definition in the attribute dictionary. Ext_id, attr_ext_name, and attr_type cannot be changed, all other parameters must be set

Parameters:

ext_id	- Extension ID of attribute definition to be modified		IN
attr_name	- description for attribute definition to be modified		IN
attr_flags	- SIMONE_NO_FLAG	attribute hidden for SIMONE GUI, attribute for internal SIMONE usage do not modify	IN
	. SIMONE_FLAG_VISIBLE	Visible and can be modified by user	
attr_mask	- SIMONE_OBJTYPE_CS SIMONE_OBJTYPE_CV SIMONE_OBJTYPE_PIPE SIMONE_OBJTYPE_RECP SIMONE_OBJTYPE_VA SIMONE_OBJTYPE_SE SIMONE_OBJTYPE_RE SIMONE_OBJTYPE_NS SIMONE_OBJTYPE_NO ..SIMONE_OBJTYPE_MS SIMONE_OBJTYPE_NRV SIMONE_OBJTYPE_SUB SIMONE_OBJTYPE_UNIT		IN
default_value	- default_value for attribute definition to be modified or removed empty string or NULL: default for values is removed		IN
lower_limit	- lower_limit for attribute definition to be modified or removed empty string or NULL: lower limit for values is removed		IN
upper_limit	- upper_limit for attribute definition to be modified or removed empty string or NULL: upper limit for values is removed		IN
flags	- reserved for future use		IN

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

Return values:

simone_status_nolicense	no license
simone_status_badseq	no network selected attribute values cannot be written, if an edited version of the network exists
simone_status_badpar	invalid parameter
simone_status_invid	extension ID is not an attribute extension
simone_status_locked	attributes are already in use by another application
simone_status_ok	ok

12.3 simone_delete_attr_def(ext_id, flags)

Delete an attribute definition from the attribute dictionary.

Parameters:

ext_id	- ID of attribute extension of attribute definition to be deleted	IN
flags	- reserved for future use	IN

Return values:

simone_status_nolicense	no license
simone_status_badseq	no network selected attribute values cannot be deleted, if an edited version of the network exists
simone_status_invid	extension ID is not an attribute extension
simone_status_locked	attributes are already in use by another application
simone_status_ok	ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

12.4 simone_get_first_attr_def(ext_id, flags)

Get first attribute definition from the attribute dictionary.

Parameters:

ext_id	- ID of attribute extension for attribute value	OUT
flags	- reserved for future use	IN

Return values:

simone_status_nolicense	no license
simone_status_badseq	no network selected attribute values cannot be written, if an edited version of the network exists
simone_status_locked	attributes are already in use by another application
simone_status_not_found	no attribute definitions found
simone_status_ok	ok

12.5 simone_get_next_attr_def(ext_id, flags)

Get next attribute definition from the attribute dictionary.

Parameters:

ext_id	- ID of attribute extension for attribute value	OUT
flags	- reserved for future use	IN

Return values:

simone_status_nolicense	no license
simone_status_badseq	no network selected attribute values cannot be written, if an edited version of the network exists
simone_status_locked	attributes are already in use by another application
simone_status_not_found	no more attribute definitions found
simone_status_ok	ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

12.6 simone_attr_def_info(ext_id, attr_name, attr_name_len, attr_ext_name, ext_name_len, attr_flags, attr_mask, attr_type, default_value, default_len, lower_limit, lower_len, upper_limit, upper_len, flags)

Get data of attribute definition from the attribute dictionary.

Parameters:

ext_id	- Extension ID of attribute definition		IN
attr_name	- description of attribute definition		OUT
attr_name_len	- length including terminating 0 for description of attribute definition		IN
attr_ext_name	- extension of attribute definition		OUT
ext_name_len	- length including terminating 0 for extension of attribute definition		IN
attr_flags	- SIMONE_NO_FLAG	attribute hidden for SIMONE GUI, attribute for internal SIMONE usage do not modify	OUT
	. SIMONE_FLAG_VISIBLE	Visible and can be modified by user	
attr_mask	- SIMONE_OBJTYPE_CS		OUT
	SIMONE_OBJTYPE_CV		
	SIMONE_OBJTYPE_PIPE		
	SIMONE_OBJTYPE_RECP		
	SIMONE_OBJTYPE_VA		
	SIMONE_OBJTYPE_SE		
	SIMONE_OBJTYPE_RE		
	SIMONE_OBJTYPE_NS		
	SIMONE_OBJTYPE_NO		
	..SIMONE_OBJTYPE_MS		
	SIMONE_OBJTYPE_NRV		
	SIMONE_OBJTYPE_SUB		
	SIMONE_OBJTYPE_UNIT		
attr_type	- SIMONE_DATA_INT		OUT
	SIMONE_DATA_FLOAT		
	SIMONE_DATA_STRING		
	SIMONE_DATA_BOOL		
	SIMONE_DATA_DATETIME		
default_value	- default_value of attribute definition		OUT
	NULL: do not return a default_value		
default_len	- length including terminating 0 for default_value		IN
	< 0: do not return a default_value		
lower_limit	- lower_limit of attribute definition		OUT
	NULL: do not return a lower_limit		
lower_len	- length including terminating 0 for lower_limit		IN
	< 0: do not return a lower_limit		

upper_limit	- upper_limit of attribute definition NULL: do not return an upper_limit	OUT
upper_len	- length including terminating 0 for upper_limit < 0: do not return an upper_limit	IN
flags	- reserved for future use	IN

Return values:

simone_status_nolicense	no license
simone_status_badseq	no network selected attribute values cannot be written, if an edited version of the network exists
simone_status_invid	extension ID is not an attribute extension
simone_status_locked	attributes are already in use by another application
simone_status_badpar	invalid parameter
simone_status_ok	ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

12.7 simone_write_attr(obj_id, ext_id, value_str)

Write a value for a user defined attribute. The attribute must be defined before in the attribute dictionary.

Parameters:

obj_id	- ID of object for attribute value to be set for	IN
ext_id	- ID of attribute extension for attribute value	IN
value_str	- string with attribute value contents of the value string must match the defined attribute type: Integer Float String Boolean "True" | "False" | "1" | "0" | "1.0" | "0.0" Date string "<day>.<month>.<year> <hour>:<minute>" if year < 1970, add 100 to years (1923 ==> 2023) 1.1.1970 0:00 (UTC) <= allowed date	IN

Return values:

simone_status_nolicense	no license
simone_status_badseq	no network selected attribute values cannot be written, if an edited version of the network exists
simone_status_invid	extension ID is not an attribute extension or object-ID invalid
simone_status_badpar	attr_value_str = NULL or empty attribute value does not fit to limits in attribute definition wrong date string
simone_status_locked	attributes are already in use by another application
simone_status_ok	ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

12.8 simone_delete_attr_val(obj_id, ext_id, flags)

Delete a value for a user defined attribute.

Parameters:

obj_id	- ID of object for attribute value to be set for	IN
ext_id	- ID of attribute extension for attribute value	IN
flags	- reserved for future use	IN

Return values:

simone_status_nolicense	no license
simone_status_badseq	no network selected attribute values cannot be written, if an edited version of the network exists
simone_status_invid	extension ID is not an attribute extension or object-ID invalid
simone_status_locked	attributes are already in use by another application
simone_status_ok	ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

12.9 simone_start_write_attr(flags)

This routine opens a transaction bracket for subsequent calls to *simone_write_attr*, which should be closed by *simone_end_write_attr*.
To do this might significantly speed up the execution of the writing calls. If there are many.

Parameters:

flags	- flags reserved for future use	IN

Return values:

simone_status_nolicense	no license
simone_status_badseq	SIMONE API not initialized, no network selected
simone_status_ok	ok

12.10 simone_end_write_attr(flags)

This routine closes the transaction bracket for calls to *simone_write_attr*.

13 Load Profiles and Functions

Load profiles are used to define patterns of boundary condition parameters (supply or offtake quantities). They are edited in a dialogue at the user interface and can be referenced repeatedly for different nodes, using different multiplication factors to assign different levels of load of a similar pattern at different locations.

The API provides interfaces to support creating such load profiles, and returns ids that are used to create entries in a scenario by *simone_write_ex*, that reference these profiles.
As load profiles belong to the scenario for which they are defined, a scenario must be open in CREATE or WRITE mode to write profiles.

Profiles are defined by a series of values at consecutive times and at equal time steps.
To create a profile, use the following sequence:

simone_begin_profile()	create new profile or overwrite existing profile
simone_write_profile()	create or overwrite a value for one time step
simone_end_profile()	finish to write values for one profile

The *simone_write_profile* function works on a temporary buffer, all profile data are written to the scenario but with the *simone_end_profile* function.

If several profiles are created for a scenario, the greatest common divisor of their time steps is used to create a common time step. For the respective steps input data for the simulation will be generated at the time a scenario is being calculated. If for a particular profile no value is defined for a time step, a value will be interpolated according to the interpolation type selected for the profile.

The sequence of a profile's values can be read by:

simone_begin_read_profile()	set prf_id of existing profile before reading profile values
simone_read_profile()	read value of one time step

The *simone_profile_in_use* function checks, if a profile is used to define any boundary flow in the current scenartio.
To delete, rename or update a profile use *simone_delete_profile*, *simone_rename_profile* or *simone_update_profile* respectively.
Functions are defined in a dialogue at the user interface. The API provides interfaces to get the id's of already defined functions (*simone_get_function*) as well as to re-define them or even create new functions (*simone_define_function*). The interfaces always return id's that are used to reference them when creating scenario entries by *simone_write_ex*.

Functions may be local, i.e. stored for the current scenario only, or global, i.e. stored for the network. They may be defined by arithmetic or logical expressions and can be used to define 'user-defined extensions'. For more detail see the respective chapter in the userguide.

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

13.1 simone_begin_profile(name, interpolation, prf_id)

This function initiates writing a profile *name.*

If the profile already exists, the matching *prf_id* is returned. If the profile does not yet exist, a new profile entry is created and it's *prf_id* is returned. If the name is already in use for another type of entry in the actual scenario, an error is returned, because names for local functions, load profiles, macros and source names must be unique.

When profile values are needed at run-time in between time steps, these are interpolated either linearly or 'stepwise', i.e. the next value in time is used. This interpolation behavior can be defined by *interpolation*.

Parameters:

name	- name of load profile NULL or empty string: use prf_id (replace values)	IN
interpolation	- SIMONE_LINEAR_INTERPOLATION (default) SIMONE_STEPWISE_INTERPOLATION interpreted only when creating a new profile	IN
prf_id	- ID of profile 0: create new profile, request by name of profile existing prf_id: request by prf_id (replace values)	IN
	- ID of profile	OUT

Return values:

simone_status_nolicense	no license
simone_status_nofile	no scenario open
simone_status_badseq	no network selected scenario only open for reading profile already in use
simone_status_badpar	name = NULL or empty name already in use for local function, macro, source name
simone_status_error	internal error
simone_status_ok	ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

13.2 simone_write_profile(prf_id, atime, value)

Create or overwrite a profile value for a time slot.

The *simone_write_profile* function works on a temporary buffer, all profile data are written to the scenario with the *simone_end_profile* function.

Parameters:

prf_id	- ID of profile	IN
atime	- time slot	IN
value	- profile value for time slot	IN

Return values:

simone_status_nolicense	no license
simone_status_nofile	no scenario open
simone_status_badseq	no network selected scenario only open for reading no call of *simone_begin_profile* before invalid prf_id
simone_status_invtime	invalid time for profile entry
simone_status_ok	ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

13.3 simone_end_profile(prf_id)

Finish writing values to requested profile.

The *simone_write_profile* function works on a temporary buffer, all profile data are written to the scenario with the *simone_end_profile* function.

Parameters:

prf_id	- ID of profile	IN

Return values:

simone_status_nolicense	no license
simone_status_nofile	scenario not open
simone_status_badseq	no network selected scenario only open for reading no call of *simone_begin_profile* before invalid prf_id
simone_status_ok	ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

13.4 simone_begin_read_profile(prf_id, properties, flags)

This function initiates reading profile *values.*

When profile values are needed at run-time in between time steps, these are interpolated either linearly or ‘stepwise’, i.e. the next value in time is used. This interpolation behavior is returned by the properties.

Parameters:

prf_id	- ID of profile	IN
properties	- SIMONE_LINEAR_INTERPOLATION SIMONE_STEPWISE_INTERPOLATION	OUT
flags	- reserved for future use	IN

Return values:

simone_status_nolicense	no license
simone_status_nofile	no scenario open
simone_status_badseq	no network selected
simone_status_badpar	properties = NULL invalid prf_id name already in use for local function, macro, source name
simone_status_invid	constant profile not allowed
simone_status_ok	ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

13.5 simone_read_profile(prf_id, atime, value)

Read the profile value for the next time slot.

Parameters:

prf_id	- ID of profile	OUT
atime	- time slot	OUT
value	- profile value for time slot	OUT

Return values:

simone_status_nolicense	no license
simone_status_nofile	no scenario open
simone_status_badseq	no network selected no call of *simone_begin_read_profile* before

simone_status_badpar	prf_id = NULL
	atime = NULL
	value = NULL

simone_status_not_found	no (more) entry found in scenario for actual profile
simone_status_ok	ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

13.6 simone_profile_in_use(prf_id, flags)

Check, if profile in use with any boundary flow.

Parameters:

prf_id	- ID of profile	IN
flags	- reserved for future use	IN

Return values:

simone_status_nolicense	no license
simone_status_nofile	no scenario open
simone_status_badseq	no network selected
simone_status_badpar	invalid prf_id name already in use for local function, macro, source name
simone_status_invid	constant profile not allowed
simone_status_not_found	ok, profile is not used
simone_status_ok	ok, profile in use with at least one boundary flow

13.7 simone_delete_profile(prf_id, flags)

Delete a profile.

Parameters:

prf_id	- ID of profile to be deleted	IN
flags	- reserved for future use	IN

Return values:

simone_status_nolicense	no license
simone_status_nofile	no scenario open
simone_status_badseq	no network selected
simone_status_badpar	invalid prf_id name already in use for local function, macro, source name prf_id is in use with a boundary flow
simone_status_invid	constant profile not allowed
simone_status_ok	ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

13.8 simone_rename_profile(prf_id, new_name, flags)

Renam a profile.

Parameters:

prf_id	- ID of profile to be renamed	IN
new_name	- new name for profile to be renamed	IN
flags	- reserved for future use	IN

Return values:

simone_status_nolicense	no license
simone_status_nofile	no scenario open
simone_status_badseq	no network selected
simone_status_badpar	invalid new_name invalid prf_id name already in use for local function, macro, source name
simone_status_invid	constant profile not allowed
simone_status_ok	ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

13.9 simone_update_profile(mode, prf_id, atime, value, flags)

Add, replace otr remove a single profile entry (time + value) or update properties (interpolation flags) for all entries of a profile.

Parameters:

mode	- SIMONE_WRITE_PROFILE_ENTRY add / replace a single profile entry SIMONE_REMOVE_PROFILE_ENTRY remove a single profile entry SIMONE_PROFILE_PROPERTIES update properties of all entries for one profile	IN
prf_id	- ID of profile to be accessed	IN
atime	- time stamp of profile entry not interpreted for update properties	IN
value	- new profile value when adding not interpreted for remove, update properties	IN
flags	- SIMONE_LINEAR_INTERPOLATION SIMONE_STEPWISE_INTERPOLATION not interpreted for add / replace / remove a single profile entry	IN

Return values:

simone_status_nolicense	no license
simone_status_nofile	no scenario open
simone_status_badseq	no network selected
simone_status_badpar	invalid mode invalid prf_id name already in use for local function, macro, source name
simone_status_invid	constant profile not allowed
simone_status_ok	ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

13.10 simone_define_function(mode, name, definition, func_id, func_type)

This routine defines or re-defines a function. If the function does not yet exist, a new entry is created, otherwise the existing entry is re-defined.
In any case, the expression supplied under *definition* is checked, and only for a valid expression, the function is created or modified.
Depending on mode, the function is stored locally for the actual scenario or globally for the network.
If the name supplied is already in use for another type of entry in the actual scenario, an error is returned, because names for local functions, load profiles, macros and source names must be unique.

The id of the function is returned as *func_id*, *func_type* gives an indication as to whether the function defines a user-defined extension or any other valid function.

Parameters:

mode	- choose local or global functions SIMONE_LOCAL_FUNCTION (default) SIMONE_GLOBAL_FUNCTION is not interpreted, if function is identified by a valid func_id	IN
name	- name of function NULL or empty: if func_id existing, replace definition only interpreted, if function is to be created (func_id = 0)	IN
definition	- function definition string	IN
func_id	- ID of function 0: create a new function existing func_id: replace definition	IN
	- ID of function	OUT
func_type	- type of function SIMONE_USER_DEFINED_EXTENSION SIMONE_VALID_FUNCTION SIMONE_GLOBAL_USER_DEFINED SIMONE_GLOBAL_VALID_FUNCTION	OUT

Return values:

simone_status_nolicense	no license
simone_status_nofile	no scenario open
simone_status_badseq	no network selected scenario only open for reading
simone_status_badpar	invalid function definition name already in use for a load profile, macro, or source name
simone_status_locked	global object for the actual network is currently being edited by another user
simone_status_ok	ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

13.11 simone_define_function_ex(mode, name, definition, unit, comment, category, func_id, func_type, flags)

The same as *simone_define_function*, but allowing to also specify unit, comment and category of the function.

Note: Either a unit descriptor containing a unit type and a unit code or simply containing a unit code can be set for a unit definition of the function. If only the unit_type is set, the default unit for this type is used. For details refer to *simone_unit2des* in the chapter Unit Handling.

Parameters:

mode	- choose local, global or universal function SIMONE_LOCAL_FUNCTION (default) SIMONE_GLOBAL_FUNCTION is not interpreted, if function is identified by a valid func_id	IN
name	- name of function NULL or empty: if func_id existing, replace definition only interpreted, if function is to be created (func_id = 0)	IN
definition	- function definition string	IN
unit	- unit descriptor specifying the unit or SIMONE_UNIT_DEFAULT	IN
comment	- comment for function, fill with blanks to delete comment	IN
category	- category of function, fill with blanks to delete category	IN
func_id	- ID of function 0: create a new function existing func_id: replace definition, unit_type, unit, comment, category - ID of function	IN OUT
func_type	- type of function SIMONE_USER_DEFINED_EXTENSION SIMONE_VALID_FUNCTION SIMONE_GLOBAL_USER_DEFINED SIMONE_GLOBAL_VALID_FUNCTION SIMONE_UNIVERSAL_USER_DEFINED SIMONE_UNIVERSAL_VALID_FUNCTION	OUT
flags	- reserved for future use	IN

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

Return values:

simone_status_nolicense	no license
simone_status_nofile	no scenario open
simone_status_badseq	no network selected scenario only open for reading
simone_status_nofile	no scenario open and local function requested
simone_status_badpar	invalid function name and invalid func_id name already in use for a load profile, macro, source name or a function with different mode invalid mode invalid function definition invalid func_type
simone_status_locked	global object for the actual network is currently being edited by another user
simone_status_not_found	requested function name or function id not found
simone_status_inv_definition	invalid function definition string
simone_status_ok	ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

13.12 simone_get_function(func_id, name, name_len, definition, definition_len, func_type)

This interface allows to retrieve the id or definition of an existing function.
The returned function type may also be bit-tested, i.e. a logical AND of the returned function type ‘with SIMONE_USER_DEFINED, is unequal to zero regardless of whether it is a local or global function.

Parameters:

func_id	- ID of function 0: request by name of function existing func_id: request by func_id	IN
	- ID of function	OUT
name	- name of function NULL or empty string: request by func_id valid name: request by name of function	IN
	- name of function	OUT
name_len	- max. length of name of function (interpreted only if request by func_id) 0: do not return name of function	IN
definition	- function definition NULL or empty string: do not return definition	OUT
definition_len	- length of definition of function 0: do not return definition	IN
func_type	- type of function SIMONE_USER_DEFINED_EXTENSION (local) SIMONE_VALID_FUNCTION SIMONE_GLOBAL_USER_DEFINED (global) SIMONE_GLOBAL_VALID_FUNCTION	OUT

Return values:

simone_status_nolicense	no license
simone_status_badseq	no network selected
simone_status_badpar	name = NULL or empty and func_id = 0 name already in use for a load profile, macro, or source name
simone_status_not_found	requested function name or id not found
simone_status_ok	ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

13.13 simone_get_function_ex(func_id, name, name_len, definition, definition_len, unit, comment, comment_len, category, category_len, func_type, flags)

This interface allows to retrieve the id or definition of an existing function.
The returned function type may also be bit-tested, i.e. a logical AND of the returned function type ‘with SIMONE_USER_DEFINED, is unequal to zero regardless of whether it is a local or global function.

Parameters:

func_id	- ID of function 0: request by name of function existing func_id: request by func_id		IN
	- ID of function		OUT
name	- name of function NULL or empty string: request by func_id valid name: request by name of function		IN
	- name of function		OUT
name_len	- max. length of name of function (interpreted only if request by func_id)		IN
definition	- function definition		OUT
definition_len	- length of definition of function		IN
unit	- unit descriptor specifying the unit or SIMONE_UNIT_DEFAULT		OUT
comment	- comment for function		OUT
comment _len	- length of comment of function		IN
category	- category of function		OUT
category _len	- length of category of function		IN
func_type	- type of function SIMONE_USER_DEFINED_EXTENSION SIMONE_VALID_FUNCTION SIMONE_GLOBAL_USER_DEFINED SIMONE_GLOBAL_VALID_FUNCTION SIMONE_UNIVERSAL_USER_DEFINED SIMONE_UNIVERSAL_VALID_FUNCTION	 (local) (global) (universal) 	OUT
flags	- for future use		IN

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

Return values:

simone_status_nolicense	no license
simone_status_badseq	no network selected
simone_status_nofile	no scenario open and local function requested
simone_status_badpar	invalid function name and invalid func_id invalid unit_type invalid unit invalid func_type return buffers too small (name_len, definition_len, comment_len, category_len)
simone_status_not_found	requested function name or id not found
simone_status_ok	ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

13.14 simone_remove_function(func_id, flags)

This interface allows to remove an existing function.

Parameters:

func_id	- ID of function	IN
flags	- for future use	IN

Return values:

simone_status_nolicense	no license
simone_status_badseq	no network selected
simone_status_nofile	no scenario open and local function requested
simone_status_badpar	invalid func_id
simone_status_locked	global object is currently being edited by another user
simone_status_not_found	func id not found
simone_status_ok	ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

13.15 simone_rename_function(func_id, func_name, flags)

This interface allows to rename an existing function.

Parameters:

func_id	- ID of function	IN
name	- new name for function	IN
flags	- for future use	IN

Return values:

simone_status_nolicense	no license
simone_status_badseq	no network selected
simone_status_nofile	no scenario open and local function requested
simone_status_badpar	invalid func_id invalid new name for function name already in use for a load profile, macro, or source name
simone_status_locked	global object is currently being edited by another user
simone_status_ok	ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

14 Object sets

The SIMONE user-interface allows to create named sets of objects, which can be used at various places – see the SIMONE userguide for more detail. The following interfaces are provided to enable creating and using object sets also at the API application level.The function *simone_create_object_set* will create an empty set under a given name, to which objects can be added by *simone_add_to_object_*set. Like at the user interface, a set representing a path connecting the already defined objects can be generated using *simone_make_path_from_object_set*.
Also the 'flood' function of the user interface is available as *simone_flood_area*.
The objects in a set can be inquired by *simone_get_next_id_from_set* and *simone_delete_object_set* allows to remove a set as a whole.

Use simone_varid() to get the id of an already existing object set.

NOTE: Some interface functions may return a simone_status_locked error as object sets belong to the network global objects (like global functions or macros), which can only be manipulated by a single user at a time.

14.1 simone_create_object_set(obj_set_name, obj_set_id, flags)

This function creates a new empty object set in the current network under the specified name.

Parameters:

obj_set_name	- name for new object set to be created		IN
obj_set_id	- identifier of created object set		OUT
flags	SIMONE_FLAG_TEMPORARY	make temporary object set, contents of object set cannot be seen by SIMONE GUI	IN

Return values:

simone_status_nolicense	no license
simone_status_badseq	no network selected
simone_status_locked	global objects currently being edited by another user
simone_status_badpar	name = NULL or empty or name already in use for another global object
simone_status_ok	ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

14.2 simone_delete_object_set(obj_set_id, flags)

This function deletes an existing object set.

Parameters:

obj_set_id	- identifier of an existing object set	IN
flags	- reserved for future use	IN

Return values:

simone_status_nolicense	no license
simone_status_badseq	no network selected
simone_status_locked	global objects currently being edited by another user
simone_status_badpar	obj_set_id is invalid
simone_status_ok	ok

14.3 simone_add_to_object_set(obj_set_id, obj_id, flags)

This function adds an object to an existing object set.
It does not allow for duplicates, i.e., if an object is already in the set, the attempt to add it a second time is silently ignored.

Parameters:

obj_set_id	- identifier of an existing object set	IN
obj_id	- identifier of object to be added	IN
flags	- reserved for future use	IN

Return values:

simone_status_nolicense	no license
simone_status_badseq	no network selected
simone_status_locked	global objects currently being edited by another user
simone_status_badpar	obj_set_id or obj_id is invalid
simone_status_ok	ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

14.4 simone_make_path_from_object_set(obj_set_id, slip_factor, flags)

This function converts the given object set – if possible - to a set representing a path or route connecting the objects of the original set.
If the pig tracking option is licensed, specifying a slip factor marks the resulting set to represent a pig.

Parameters:

obj_set_id	- identifier of an existing object set		IN
slip_factor	- slip factor for a pig path or pig route, a pig path or route is created, if the slip_factor > 0		IN
flags	- SIMONE_MAKE_PATH |	create and store a path	IN
	SIMONE_MAKE_ROUTE	create and store a route	

Return values:

simone_status_nolicense	no license
simone_status_badseq	no network selected
simone_status_locked	global objects currently being edited by another user
simone_status_badpar	obj_set_id is invalid
simone_status_failed	no route or path found for object set
simone_status_ok	Ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

14.5 simone_flood_area(type, obj_set_id, obj_id, flags)

This function adds elements and nodes of a 'flood' operation to an existing object set, which may be empty

Parameters:

type	- type of flood		IN
	SIMONE_TYPE_AREA	all nodes/elements in an area	
	SIMONE_TYPE_SEPARABLE_AREA	area isolated by closest CS, CV, VA	
	SIMONE_TYPE_UPSTREAM_AREA	flood upstream of node	
	SIMONE_TYPE_DOWNSTREAM_AREA	flood downstream of node	
obj_set_id	- identifier of an existing object set		IN/OUT
obj_id	- object id of flood start		IN
Flags	- flags		IN
	SIMONE_FLAG_ADD	add nodes/elements to existing object set	
	SIMONE_FLAG_REPLACE	clear object set before adding nodes/elements to object set	

Return values:

simone_status_nolicense	no license
simone_status_badseq	no network selected
simone_status_nofile	no scenario selected
simone_status_locked	global objects currently being edited by another user
simone_status_badpar	obj_set_id is invalid, obj_id is invalid wrong type of ob_id
simone_status_ok	Ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

14.6 simone_get_next_id_from_set(obj_set_id, obj_id, flags)

This function retrieves the next object id stored with the specified object set. The next object id is returned. If the first object id is required, then obj_id=0 is to be supplied.
The names and types of the returned object is's can be inquired by *simone_id2name*.

Parameters:

obj_set_id	- identifier of an existing object set	IN
obj_id	- identifier of object to be added	IN/OUT
flags	- reserved for future use	IN

Return values:

simone_status_nolicense	no license
simone_status_badseq	no network selected
simone_status_badpar	obj_set_id or obj_id is invalid
simone_status_not_found	ok, no (more) object id found
simone_status_ok	Ok

14.7 simone_set_object_set_category(object_set_id, category)

Allows to set a category for an existing object set.

Parameters:

obj_set_id	- id of an existing object set	IN
category	- category of object sets, maybe used for filtering NULL or empty string: delete category, if existing	IN

Return values:

simone_status_badseq	API is not initialized or no network selected
simone_status_nolicence	no license
simone_status_locked	global objects currently being edited by other user(s)
simone_status_not_found	invalid obj_set_id =>, id does not exist
simone_status_badpar	invalid object set id, id is already in use for another global object or

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

length of category is too large or
universal object sets must not be changed

14.8 simone_get_object_set_catagory(obj_set_id, category, category_max_len)

Retrieves the category of an existing object set.

Parameters:

obj_set_id	- id of an existing object set	IN
category	- category of object sets, maybe used for filtering	IN
category_max_len	- length of supplied buffer 'category' including terminating '\0'	IN

Return values:

simone_status_badseq	API is not initialized or no network selected
simone_status_nolicence	no license
simone_status_locked	global objects currently being edited by other user(s)
simone_status_not_found	invalid obj_set_id =>, id does not exist
simone_status_badpar	invalid object set id, id is already in use for another global object or length of category is too small or 'category' == NULL

14.9 simone_create_object_set_ex(obj_set_name, obj_set_definition, obj_set_id, flags)

Creates a new object set based on a „definition string".

Parameters:

obj_set_name	- name for new object set to be created	IN
obj_set_definition	- string to define the content of the object set to be created. "*" -> add all objects "*:<OBJTYPE>[,<OBJTYPE>[,<objtype>]].." whould add all objects of specified types PI,VA,CS,CV,RE,NRV,MS,SUB,..	IN
obj_set_id	- id of created object set	OUT
flags	- SIMONE_FLAG_TEMPORARY: create temporary object set, cannot be seen by GUI	IN

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

Return values;

simone_status_badseq	API is not initialized
simone_status_nolicence	no license
simone_status_badseq	no network selected
simone_status_locked	global objects currently being edited by another user
simone_status_badpar	obj_set_name is already in use for another global object
simone_status_ok	ok

14.10 simone_object_set_filter(obj_set_id, filter_expression, flags)

Allows to modify the supplied set according to the supplied 'filter' expression.

Paraameters:

obj_set_id	- id of object set to be modified	IN
filter_expression	- expression to filter (reduce) the objects in the supplied object set: [<obj-type-list>]['/'<expression>] where <obj-type-list> = <OBJTYPE>[,<OBJTYPE>[,<objtype>]].. <OBJTYPE> = PI,VA,CS,CV,RE,NRV,MS,SUB,.. <expression> = a valid SIMONE expression like (%.MODE=="BP") or (%.Q!=0)||(TFNUL(%.QP)!=0)	IN
Flags	- reserved for futrure use. supply SIMONE_NO_FLAG	IN

Return values:

simone_status_badseq	SIMONE API is not initialized
simone_status_nolicence	no license
simone_status_badseq	no network selected
simone_status_locked	global objects currently being edited by another user
simone_status_badpar	filter_expression is invalid
simone_status_ok	ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

15 Calculation, Status and Messages

A scenario that has been defined using the API or at the user interface may be calculated using the function *simone_execute* or *simone_execute_ex*.

The results and messages from the execution can be visualized at the user interface as well as being retrieved by the API.

Generally, the calculation status of a scenario can be retrieved by *simone_calculation_status,* regardless whether it has been created or calculated with the interactive user interface, the online environment or with the API. Once it has been calculated, success and/or error messages of the calculation can be read using *simone_get_first_message* and *simone_get_next_message*. If needed, filter conditions can be set for checking only a special error message or for checking only messages which refer to a special object.

15.1 simone_execute(status_txt, txt_len)

The scenario is being calculated.

The scenario must be open, the type of the scenario (runtype), the name of the initial conditions as well as the time interval are used as stored for the scenario. The initial conditions must exist.
Run types are supported according to the licensed options of the particular installation.
After execution, the scenario is open in READ mode and status, messages and results can be retrieved. For convenience, this routine already returns the calculation status the same way as *simone_calculation_status* does.
Unlike the SIMONE GUI this function works in the background and does not display any window. If a window containing a message box with calculation messages, calculation status, remove and cancel button is required, use the *simone_execute_ex* function.

Note:
This function requires an extended API license

Parameters:

status_txt	- status of calculation “RUNOK”: successful calculation all other status texts indicate an incomplete or failed calculation	OUT
txt_len	- maximum length of status	IN

Return values:

simone_status_nolicense	no license
simone_status_badseq	no network selected
simone_status_nofile	no scenario open
simone_status_locked	scenario already locked by another instance
simone_status_badpar	wrong or invalid scenario properties
simone_status_failed	incomplete or failed calculation
simone_status_ok	Ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

15.2 simone_execute_ex(status_txt, txt_len, flags)

The scenario is being calculated as with the *simone_execute* function. Depending on the flag this function works silently in the background like the *simone_execute* function (SIMONE_NO_FLAG) or it shows a message box is like the SIMONE GUI does (SIMONE_FLAG_INTERACTIVE_MSG).

The scenario must be open, the type of the scenario (runtype), the name of the initial conditions as well as the time interval are used as stored for the scenario. The initial conditions must exist.
Run types are supported according to the licensed options of the particular installation.

If the SIMONE_FLAG_INTERACTIVE_MSG is set, during the calculation of the scenario results a window is displayed containing a message box with execution messages, ok button, remove messages button and a cancel button. The user must quit this window after the calculation is finished or the the user can interrupt the the calculation of the scenario results by clicking the cancel button.

After execution, the scenario is open in READ mode and status, messages and results can be retrieved. For convenience, this routine already returns the calculation status the same way as *simone_calculation_status* does.

If SIMONE_NO_FLAG is set, this interface works identical to *simone_execute*.

Note:
This function requires an extended API license

Parameters:

status_txt	- status of calculation “RUNOK”: successful calculation all other status texts indicate an incomplete or failed calculation	OUT
txt_len	- maximum length of status	IN
flags	- flags for execution SIMONE_NO_FLAG | background, no window SIMONE_FLAG_INTERACTIVE_MSG | window with message box and ok button, cancel button, remove messages button SIMONE_FLAG_EXECUTE_SCENARIO_CHAIN if set and the initial conditions are not existing, calculate all necessary scenarios to create the initial conditions.	IN

Return values:

simone_status_nolicense	no license
simone_status_badseq	no network selected
simone_status_nofile	no scenario open
simone_status_locked	scenario already locked by another instance
simone_status_badpar	wrong or invalid scenario properties
simone_status_failed	incomplete or failed calculation

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

simone_status_ok	Ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

15.3 simone_calculation_status(status_txt, txt_len)

Returns the calculation status of a scenario. The scenario must be opened for reading after calculation.

Parameters:

status_txt	- status of calculation "RUNOK": successful calculation all other status texts indicate an incomplete or failed calculation	OUT
txt_len	- maximum length of status	IN

Return values:

simone_status_nolicense	no license
simone_status_badseq	no network selected
simone_status_nofile	scenario not open undefined execution (message file not found)
simone_status_badpar	status_txt is not a valid pointer status text length too short
simone_status_noval	improperly terminated execution
simone_status_ok	Ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

15.4 simone_calculation_status_ex(scenario, status_txt, txt_len, flags)

Returns the calculation status of a scenario. The scenario must not be open.

Parameters:

scenario	- name of scenario	IN
status_txt	- status of calculation “RUNOK”: successful calculation all other status texts indicate an incomplete or failed calculation	OUT
txt_len	- maximum length of status	IN
flags	- reserved for future use	IN

Return values:

simone_status_nolicense	no license
simone_status_badseq	no network selected scenario already open
simone_status_locked	scenario is already locked by another instance
simone_status_nofile	undefined execution (message file not found)
simone_status_badpar	scenario is not a valid pointer status_txt is not a valid pointer status text length too short
simone_status_noval	improperly terminated execution
simone_status_ok	Ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

15.5 simone_set_message_filter(obj_name, msg_name)

Set filter conditions for subsequent call(s) of *simone_get_first_message* and *simone_get_next_mesage* functions. Only messages with the defined name and/or which refer to the specified object are reported.

Some messages of interest are listed here.

Name	Object available	Description
msg_qsim_abs_pmin	yes	absolute minimum of pressure reached (probably intake/offtake disbalance)
msg_qsim_abs_pmax	yes	absolute maximum of pressure reached (probably intake/offtake disbalance)
is$run18	yes	pressure deviation (reconstruction only)
msg_ext_alarm	yes	Leak alarm (leak detection only)
msg_hfr_HFRwarning	-	Hydrate Formation possible in network.
msg_hfr_DEWwarning	-	Free water may occur in network.

Parameters:

obj_name	- object which is referred in message NULL or empty string: all objects are chosen (reset filter)	IN
msg_name	- message name to be reported NULL or empty string: all messages are chosen (reset filter)	IN

Return values:

simone_status_nolicense	no license
simone_status_badseq	no network selected
simone_status_nofile	scenario not open
simone_status_badpar	object name too long message name does not exist
simone_status_ok	ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

15.6 simone_get_first_message(msg, msg_len, msg_time, severity, obj_name, obj_len, msg_name, msg_name_len)

Read first status message from last calculation of a scenario. If desired, filter conditions may be set before with the *simone_set_message_filter* function. Settings of the filter conditions keep valid for subsequent call(s) of *simone_get_next_message*.The scenario must be opened for reading after calculation. All string outputs can be deselected.

Parameters:

msg	- status message to be returned NULL: do not return message	OUT
msg_len	- maximum length of status message to be returned <= 0: do not return message	IN
msg_time	- time stamp of status message -1: message has no time stamp	OUT
severity	- reserved for future use	OUT
obj_name	- name of referred object in status message, empty if no object referred - NULL: do not return object	OUT IN
obj_len	- maximum length of name of referred object <= 0: do not return object	IN
msg_name	- message name - NULL: do not return message name	OUT IN
msg_name_len	- maximum length of message name <= 0: do not return message name	IN

Return values:

simone_status_nolicense	no license
simone_status_badseq	no network selected
simone_status_nofile	scenario not open undefined execution (message file not found)
simone_status_not_found	no (matching) message
simone_status_ok	ok, message returned

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

15.7 simone_get_next_message(msg, msg_len, msg_time, severity, obj_name, obj_len, msg_name, msg_name_len)

Read next status message from last calculation of a scenario. The scenario must be opened for reading after calculation and *simone_get_first_message* must be called first. All string outputs can be deselected.

Parameters:

msg	- status message to be returned NULL: do not return message	OUT
msg_len	- maximum length of status message to be returned <= 0: do not return message	IN
msg_time	- time stamp of status message -1: message has no time stamp	OUT
severity	- reserved for future use	OUT
obj_name	- name of referred object in status message, empty if no object referred - NULL: do not return object	OUT IN
obj_len	- maximum length of name of referred object <= 0: do not return object	IN
msg_name	- message name - NULL: do not return message name	OUT IN
msg_name_len	- maximum length of message name <= 0: do not return message name	IN

Return values:

simone_status_nolicense	no license
simone_status_badseq	no network selected simone_get_first_message() must be called before
simone_status_nofile	scenario not open undefined execution (message file not found)
simone_status_not_found	no more (matching) messages
simone_status_ok	ok, message returned

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

15.8 simone_write_message(status_code, category, message, flags)

This interface allows to write a message into the SIMONE Message Window or the SIMONE Cycle Control log window, depending on whether the API application is run interactively or in the context of the SIMONE Cycle Control.
Parameters:

status_code	- integer number out of 0 | 1 | [8000..8999] 0 = SIMONE_STATUS_NORMAL: message is shown in black color 1 = SIMONE_STATUS_WARNING: warning or infomation is shown in ocker color 8000..8999 = SIMONE_STATUS_ERR_BASE + x: error is shown in red color	IN
Category	- string describing the originator of the message. This string is shown in the column 'Category' of the Cycle Controls log window. if NULL is supplied the name of the calling program is used as category string (without path)	IN
Message	- message to be shown	IN
Flags	- reserved for future use	IN

Return values:

simone_status_nolicense	no license
simone_status_badseq	SIMONE API not initialized
simone_status_ badpar	status_code out of range
simone_status_error	failed to send message via network
simone_status_ok	Ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

16 Time conversion

Times at the user interface or in the configuration are handled in text format.
SIMONE internally calculates float times according to midnight of the start of the scenario (00:00 at the day of initime.
The following functions convert times between these formats and the ANSI C format (time_t, see also the note in the section time handling).

Note:
The SIMONE float time and the times in text format used by the user interface always count the hours from midnight at the day of initime, so the float times can pass over 24.0, what is reflected by a leading number of days separated by "\" from the time of day in the text format at the user interface.

At days switching the daylight saving time, the SIMONE float times differ from the clock, because those days lose one hour or get an additional hour.

16.1 simone_time_ansi2simone(atime, initime)

Conversion from ANSI C time to SIMONE float time according to midnight of the start day of the scenario

Parameters:

atime	- time to be converted in ANSI C format (time_t)	IN
initime	- start of scenario in ANSI C format (time_t) reference date	IN

Return values:

SIMONE time in float format (hours counted from midnight(initime))

16.2 simone_time_simone2ansi(day, month, year, ftime)

Conversion from SIMONE float time to ANSI C time

Parameters:

day	- day	IN
month	- month	IN
year	- year, 2 digits or 4 digits, years less than 70 are interpreted as years after 2000.	IN
ftime	- SIMONE float time	IN

Return values:

ANSI C time (time_t)

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

16.3 simone_time_string2float(str_time)

Conversion of SIMONE time from text format to SIMONE float format

Parameter:

str_time	- time as string:	IN
	[<DD> \] <HH> [: <MM>[: <SS>]]	
	DD: Number of days since start day	(optional)
	HH: hours	
	MM: minutes	(optional)
	SS: seconds	(optional)

Return values:

SIMONE float time

16.4 simone_date_string2int(str_date, day, month, year)

Conversion of date from text format to integer format

Parameter:

str_date	- date as string	IN
	dd.mm.yy(yy) or dd-MMM-yy MMM = JAN | FEB | MAR | APR | MAY | JUN | JUL | AUG | SEP | OCT | NOV | DEC	
day		OUT
month		OUT
year		OUT

Return values:

simone_status_badpar	invalid date string
simone_status_ok	ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

16.5 simone_datetime_string2ansi(str_date, str_time)

Conversion of date and time from text format and time to ANSI C format (time_t)

Parameter:

str_date	- date as string	IN
	dd.mm.yy(yy) or dd-MMM-yy MMM = JAN | FEB | MAR | APR | MAY | JUN | JUL | AUG | SEP | OCT | NOV | DEC	
str_time	- time as string:	IN
	[<DD> \] <HH> [: <MM>[: <SS>]]	
	DD: Number of days since start day	(optional)
	HH: hours	
	MM: minutes	(optional)
	SS: seconds	(optional)
Atime	- date / time in ANSI C format (time_t)	OUT

Return values:

ANSI C time (time_t)

16.6 simone_time_float2string(ftime, str_time, str_time_len, flags)

Conversion SIMONE time from SIMONE float format to text format

Parameters:

Ftime	- time in SIMONE float format	IN
str_time	- time as string: [<DD> \] <HH> [: <MM>[: <SS>]] DD: Number of days since start day HH: hours MM: minutes	OUT
str_time_len	- max. length of time as string	IN
Flags	- format of time as string SIMONE_FLAG_HHMM: [<DD> \] <HH> : <MM>] SIMONE_FLAG_HHMMSS: [<DD> \] <HH> : <MM> [: <SS>]	IN

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

16.7 simone_time_ansi2int(atime, day, month, year, hour, minute, second)

Convert time from ANSI C format (time_t) to Integer date / time.

Parameters:

atime	- date / time in ANSI C format (time_t)	IN
day		OUT
month		OUT
year		OUT
hour		OUT
minute		OUT
second		OUT

16.8 simone_time_ int2ansi(atime, day, month, year, hour, minute, second)

Convert time from Integer date / time to ANSI C (time_t) format.

Parameters:

atime	- date / time in ANSI C format (time_t)	OUT
day		IN
month		IN
year		IN
hour		IN
minute		IN
second		IN

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

17 Configuration

The configuration file attached at the time of simone_init contains configuration information used to define settings and to communicate them among different applications.
It is structured in sections and contains named items.
The defined API interfaces normally care for handling these settings and allow querying the relevant information. In some special cases, however, it may be required to read or write such items (only if particularly instructed). Therefore the routines *simone_get_config_item* and *simone_set_config_item* are provided.

NOTE: The EXSTAT item is the only one that should be normally set by an application, as in case the application was invoked by SIMONE, this item will be checked for the value RUNOK as the final success indication. All other items need to be treated with care, and in particular the items SIMONE_ROOT, SIMONE_NETS and NETWORK should NEVER be written to.

The following entries in the logical names section of the configuration may contain useful information:

Name	Format / Valid values	Description
SIMONE_ROOT		root directory of the SIMONE installation
SIMONE_NETS		complete path to actual directory with SIMONE networks
NETWORK		name of actual network
SIMUSER		name of actual SIMONE user
EXSTAT	RUNOK, error text	Return value of all started processes
ZDTD		not used any more
ZDTD2		not used any more
DATAPREP_EXE	Program name	Standard or customer specific program for data preparation (and export) for online cycle
DATAPREP_EXPORT_EXE	Program name	Stanadard or customer specific program for data export for online cycle

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

17.1 simone_get_config_item(section, name, value, value_len)

Read entry in configuration

Parameters:

section	- section in SIMONE configuration, empty string or NULL: use default section	IN
name	- name of entry in configuration	IN
value	- value read from entry in configuration	OUT
value_len	- maximum of length for value of entry in configuration	IN

Return values:

simone_status_nolicense	no license
simone_status_badpar	no access to SIMONE configuration section in SIMONE configuration does not exist name of entry in configuration does not exist
simone_status_ok	ok

17.2 simone_set_config_item(section, name, value)

(Over)write entry in configuration

Parameters:

section	- section in SIMONE configuration, empty string or NULL: use default section	IN
name	- name of entry in configuration	IN
value	- value to be written to entry in configuration empty string removes entry from configuration	IN

Return values:

simone_status_nolicense	no license
simone_status_badpar	no access to SIMONE configuration entry in configuration is not allowed to be modified by user section not existing in configuration
simone_status_ok	Ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

18 Accessing Request Lists

The interfaces described in the following are meant to facilitate access to the ‘request files’ produced by the ‘Activate Data Definition’ function provided in an Online System using the ‘legacy method’ for integration. These are no longer necessary with the newer ‘scripting method’. See also the ‘Data Exchange’ documention for more information.
The *simone_start_req_list* and *simone_get_req_item* functions described below allow to access the data definition in terms of an item list and get the relevant information.

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

18.1 simone_start_req_list(required_type, attached_type, import_export)

If invoked within a ‘data preparation’ (or ‘data export’) program called by SIMONE in an online cycle, this function should be called with the required type set to 0. It will then return the requested type of data preparation (i.e. what the scenario to be filled with data is meant for) and attach to the respective list configured.
Otherwise the function may also be called with a definite required type and it will attempt to find an appropriate list.
The required and/or returned type can be any of the following list:

SIMONE_REQ_TYPE_RECO	Input for (cyclic) reconstruction
SIMONE_REQ_TYPE_SIMU	Input for (cyclic) look-ahead
SIMONE_REQ_TYPE_LEAK	Input for leak detection
SIMONE_REQ_TYPE_USER_1	Input of user defined simulation 1
SIMONE_REQ_TYPE_USER_2	Input of user defined simulation 2
SIMONE_REQ_TYPE_USER_3	Input of user defined simulation 3
SIMONE_REQ_TYPE_EXPORT_PRSIM	export of values from last PRSIM cycle
SIMONE_REQ_TYPE_EXPORT_ZYSIM	export of values from last ZYSIM
SIMONE_REQ_TYPE_EXPORT_LEAK	export of leak results
SIMONE_REQ_TYPE_EXPORT_USER_1	export from a user defined simulation 1
SIMONE_REQ_TYPE_EXPORT_USER_2	export from a user defined simulation 2
SIMONE_REQ_TYPE_EXPORT_USER_3	export from a user defined simulation 3

The import_export parameter is provided to ease distinguishing the type where detail is not needed.

Parameters:

required_type	- required type as above or 0		IN
attached_type	- as of above list or SIMONE_REQ_TYPE_EXPORT_USER	export from a user defined simulation	OUT
Import_export	0: Input 1: Export	Request is for importing data Request is for data export	OUT

Return values:

simone_status_nolicense	No license
simone_status_badseq	SIMONE API not initialized a network must be selected before
simone_status_badpar	Invalid required_type
simone_status_nofile	No request file found
simone_ simone_status_error	Error reading data definitions
simone_status_not_found	No data definitions for requested type, empty request file
simone_status_ok	Ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

18.2 simone_get_req_item(obj_type, obj_id, ext_id, scada_id, scada_id_len, value, flags)

This function successively reads data definition items as of the type attached by the previous call to *simone_start_req_list*.
For convenience, the simone parameter reference is already translated to the object id and extension id as necessary for subsequent calls to the API interfaces. Also ‘direct values’ (that need not be retrieved from the external system and are defined in the data definition) are returned as binary values by the interface. This situation is indicated by the flags parameter, as well as whether the parameter to be set for SIMONE is a ‘mode’ value that needs special coding (refer to the ‘data exchange’ documentation for further detail).

Parameters:

obj_type	- type of simone object as described in simone_varid_info() function		OUT
obj_id	- object id for simone parameter		OUT
ext_id	- extension id for simone parameter		OUT
scada_id	- id for scada or other external system		OUT
scada_id_len	- maximum length for scada_id (including terminating zero)		IN
Value	- value for simone parameter from request list (only if present, see flags)		OUT
Flags	- flags		OUT
	SIMONE_DIRECT_VALUE	‘direct value’ returned from request list	
	SIMONE_CTRL_MODE	Item is a control mode parameter (extension MODE)	
	SIMONE_CONST_PARAMETER	Item is a parameter constant in time that may be written once per scenario without time	

Return values:

simone_status_nolicense	No license
simone_status_badseq	SIMONE API not initialized a network must be selected before
simone_status_invid	Object in request file not recognized
simone_status_badpar	Wrong parameter scada_id_len too small
simone_status_not_found	No (more) object found in data definitions
simone_status_ok	Ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

18.3 simone_get_req_item_ex(obj_type, obj_id, ext_id, scada_id, scada_id_len, value, info, info_len, flags)

This function extends the function of *simone_get_req_item* to allow also handling of request lists that contain special items, where the simone parameter reference successively reads data definition items as of the type attached by the previous call to *simone_start_req_list*.
For convenience, the simone parameter reference is already translated (if possible) to the object id and extension id as necessary for subsequent calls to the API interfaces.
If the extension string is no standard SIMONE extension, the ext_id is set to 0, the extension is returned as info string and SIMONE_EXTENSION_UNKNOWN is set in the flags value (The return value is set to simone_status_ok).
Also 'direct values' (that need not be retrieved from the external system and are defined in the data definition) are returned as binary values by the interface. This situation is indicated by the flags parameter, as well as whether the parameter to be set for SIMONE is a 'mode' value that needs special coding (refer to the 'data exchange' documentation for further detail).

Parameters:

obj_type	- type of simone object as described in simone_varid_info() function		OUT
obj_id	- object id for simone parameter		OUT
ext_id	- extension id for simone parameter		OUT
Scada_id	- id for scada or other external system		OUT
Scada_id_len	- maximum length for scada_id (including terminating zero)		IN
Value	- value for simone parameter from request list (only if present, see flags)		OUT
Info	- info (only filled, if the extension string is no standard SIMONE extension, refer to flags) NULL or empty string: info is not retrieved		OUT
Info_len	- maximum length for info (including terminating zero) 0: info is not retrieved		IN
Flags	- flags		OUT
	SIMONE_DIRECT_VALUE	'direct value' returned from request list	
	SIMONE_CTRL_MODE	Item is a control mode parameter (extension MODE)	
	SIMONE_CONST_PARAMETER	Item is a parameter constant in time that may be written once per scenario without time	
	SIMONE_EXTENSION_UNKNOWN	The extension string is no standard SIMONE extension	

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

Return values:

simone_status_nolicense	No license
simone_status_badseq	SIMONE API not initialized a network must be selected before
simone_status_invid	Object in request file not recognized
simone_status_badpar	Wrong parameter scada_id_len too small info_len too small
simone_status_not_found	No (more) object found in data definitions
simone_status_ok	Ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

18.4 simone_get_req_times(req_type, initime, termtime, cycle_time)

If a ‘data preparation’ (or ‘data export’) program is called by SIMONE in an online cycle, also the time interval and data cycle of the request is communicated. This function returns this information according the the request type, which should be supplied as returned by the *simone_start_req_list* function.

Parameters:

req_type	- type of request as returned by *simone_start_req_list()*	IN
initime	- starttime of scenario	OUT
termtime	- endtime of scenario	OUT
cycle_time	- data cycle in seconds	OUT

Return values:

simone_status_nolicense	No license
simone_status_badseq	SIMONE API not initialized a network must be selected before information not found in environment
simone_status_badpar	Wrong parameter Invalid req_type
simone_status_invtime	start time (initime) after end time (termtime)
simone_status_ok	OK

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

19 Traces and error handling

Most of the API functions return status values according to the table given below under ‘return values’. In case an error status is returned, an error message text can be retrieved by *simone_last_error* that may give additional detail to the cause.
Even more extensive information may be traced in a log file by setting an appropriate ‘loglevel’ for the built-in logging features of SIMONE. Several log levels for different categories are supported, that can be set for each category separately. The SIMONE API uses the category ‘api’ and it’s loglevel can be set in the configuration in the section logging (logging#*.api.loglevel) to one of the supported values: DEBUG, TRACE, INFO, WARN, ERROR and FATAL. If a log level is set, all traces with this level or with a higher one are written to the log file.
The API loglevel is read from the configuration at the time of initializing the API (e.g. *simone_init_ex*), but can be also set at runtime by *simone_set_log_level*. Further interfaces to read (*simone_get_log_level*) or reset (*simone_reset_log_level*) the loglevel are available.

Note: If no dedicated ‘api’ loglevel is set, the API functions log according to the general loglevel set, which typically is set to ERROR by the installation.

Depending on the log level the following traces are written to the log file:

SIMONE_API_LOG_ERROR	If a SIMONE API function does not return successfully, an error log is generated, unless the error is considered to be not critical and worth a warning only.
SIMONE_API_LOG_WARN	If a SIMONE API function does not return successfully, a warning is generated for not critical errors, like e.g. simone_get_config_item() if entry does not exist in configuration simone_open() if requested scenario is not existing simone_varid() if name cannot be translated simone_next_rtime() if no time slot is existing after actual time slot simone_read() if read value is not normalized simone_write() if time < initime simone_execute() if no boundary condition data in interval [initime, termtime] or if no measurement data for reconstruction
SIMONE_API_LOG_INFO	The following SIMONE API functions log some information: simone_init() logs the used configuration simone_select() logs the selected network simone_open() logs the opened scenario, the open mode, and flag simone_close() logs the closed scenario
SIMONE_API_LOG_TRACE	Each call of a SIMONE API function is logged and all data successfully returned by a SIMONE API function are logged
SIMONE_API_LOG_DEBUG	Internal functions may generate debug logs

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

19.1 Return values

Status as defined in simone_api.h, simone_api.bas, simone_api.net	Value	Comment
simone_status_ok	0	ok, function successful
simone_status_badseq	1	wrong sequence
simone_status_nofile	2	network or scenario not existing
simone_status_invid	3	ID invalid
simone_status_invtime	4	timestamp invalid
simone_status_badpar	5	parameter(s) invalid
simone_status_noval	6	no valid value
simone_status_nolicense	7	no licence
simone_status_error	8	internal error
simone_status_not_found	9	ok, no matching entry found
simone_status_locked	10	error getting lock for network or scenario or another application writes to network or scenario
simone_status_nofloat	11	value cannot be represented as a float
simone_status_failed	12	calculation failed (simone_execute) execution of batch failed (SIMONE API extensions)
simone_status_exception	13	an unknown exception has occured
simone_status_remotefail	14	a remote api call has failed to contact the API Server
simone_status_not_implemented	15	call not implemented in current environment local/remote
simone_status_insuff_license	16	insufficient license level
status_no_cycle_ctrl	17	no cycle control defined
status_incompatible	18	incompatible versions
simone_status_inv_definition	19	invalid function definition
simone_status_capacity_exceeded	20	object creation failed due to capacity restrictions

19.2 simone_last_error(error_message, len)

If a function returns an error status, additional information can be retrieved by reading the last error message.

Parameters:

error_message	- last error message	OUT
len	- maximum of length for last error message	IN

Return values:

simone_status_not_found	no error message existing
simone_status_ok	ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

19.3 simone_set_log_level(log_level, log_file_size)

This routine sets log properties for the SIMONE API category ‘api’. It overrides the log level (*.api.loglevel) and the maximum log file size (*.api.logfilesize) defined in the logging section of the actual configuration.
Logs are written to the standard log file (typically <user>.LOG) in the logs directory of the SIMONE installation, unless the item *.api.logfile in the configuration specifies a dedicated file for the API.
All traces with the specified log level or higher are written to the log file. If the size of the actual log file reaches the maximum size set it is renamed and a new log file is created.
The modified log properties keep valid until modified again or until the calling program terminates.

Parameters:

log_level	- SIMONE_API_LOG_DEBUG | SIMONE_API_LOG_TRACE | SIMONE_API_LOG_INFO | SIMONE_API_LOG_WARN | SIMONE_API_LOG_ERROR | SIMONE_API_LOG_FATAL	IN
log_file_size	- maximum size of log file in KBytes, 0: do not modify actual maximum size of log file	IN

Return values:

simone_status_nolicense	no license
simone_status_badseq	SIMONE API is not initialized
simone_status_badpar	invalid log level
simone_status_ok	ok

19.4 simone_reset_log_level()

This routine resets the log properties, it replaces all actual log properties with the log properties set in the configuration.

Return values:

simone_status_nolicense	no license
simone_status_badseq	SIMONE API is not initialized
simone_status_ok	Ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

19.5 simone_get_log_level(log_level, log_file_size)

This routine reads the actual set log properties for the SIMONE API category.

Parameters:

log_level	- SIMONE_API_LOG_DEBUG | SIMONE_API_LOG_TRACE | SIMONE_API_LOG_INFO | SIMONE_API_LOG_WARN | SIMONE_API_LOG_ERROR | SIMONE_API_LOG_FATAL	OUT
log_file_size	- maximum size of log file in Kbytes	OUT

Return values:

simone_status_nolicense	no license
simone_status_badseq	SIMONE API is not initialized
simone_status_ok	Ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

19.6 simone_prepare_system_info(status_msg, msg_len, zip_file_name, file_name_len, flags)

This routine prepares a zip-file with logging and configuration information about the current SIMONE Installation, and with the current network and scenario data attached if requested. Also information about the operating system is zipped. This file is intended to be sent by e-mail along with a problem report for possible error analysis, if desired.

Parameters:

status_msg	- status information after preparing system information	OUT
msg_len	- maximum length for status_msg including terminating zero, 0: do not return status message	IN
zip_file_name	- name of prepared zip file including path	OUT
file_name_len	- maximum length for name of prepared zip file including terminating zero, 0: do not return name of prepared zip file	IN
flags	- flags SIMONE_PREPARE_NET_INFO information for network and scenario SIMONE_PREPARE_OS_INFO System information (MSInfo32) Supported only for MS-Windows SIMONE_PREPARE_ALL_INFO all information	IN

Return values:

simone_status_nolicense	no license
simone_status_badseq	SIMONE API is not initialized
simone_status_failed	errors from executed program, use simone_last_error()for detailed information
simone_status_ok	Ok

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

20 Update references

Some changes have be made since the March 2017 state of the API interface documentation, that have been incorporated in this release of the interface documentation.

- new function simone_deselect()
- new flag SIMONE_FLAG_MAKE_NETWORK_DIR for simone_change_network_dir
- new types of data preparations or data exports in simone_start_req_list(), simone_get_req_times()
- new functions simone_get_entry_set_filter(), simone_get_entry_reset_filter()
- simone_get_entry() now using filters if set
- new function simone_varid_ex()
- add new flag for simone_extname2id()
- new function simone_var2name()
- ------------
-

© Copyright LIWACOM Informationstechnik GmbH and SIMONE Research Group s.r.o.
All rights reserved

LIWACOM
Informationstechnik GmbH

P.O. Box 102415
D-45024 Essen

Phone +49 2 01 17 03 8 - 0
Fax +49 2 01 17 03 8 - 0
E-mail postmaster@liwacom.de