Alignak Documentation

Release 2.1.3

Alignak team

Apr 19, 2019

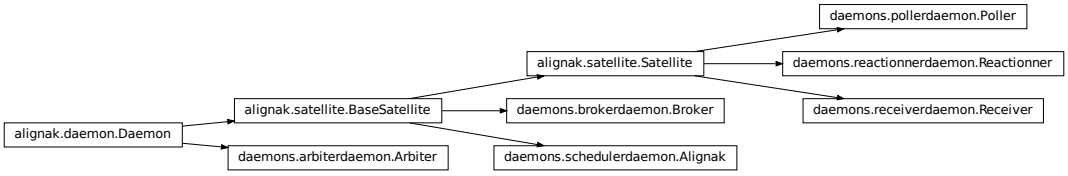
Contents

Contents:

CHAPTER 1

Diagrams

Graph Daemon :

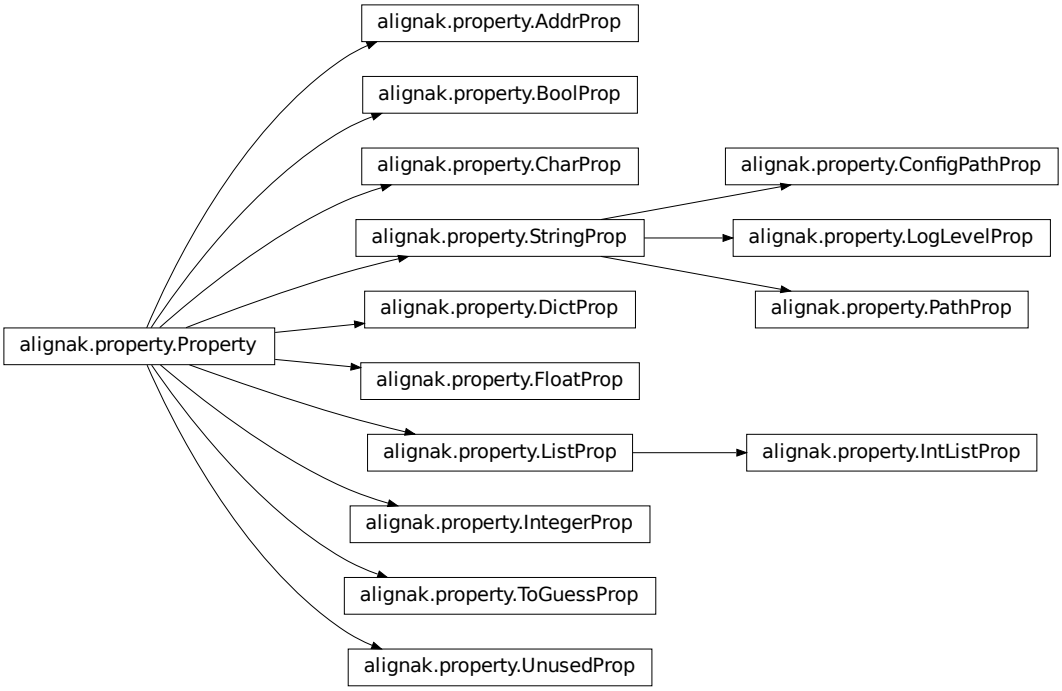


Graph Exception :

Graph Items :

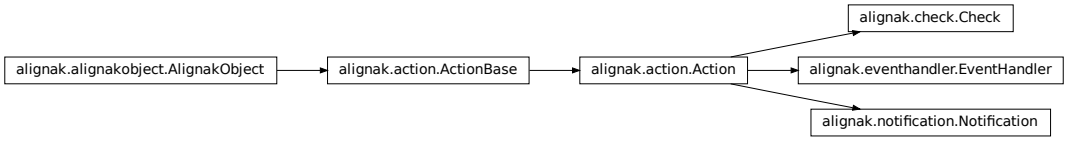
Graph Item :

Graph Property :

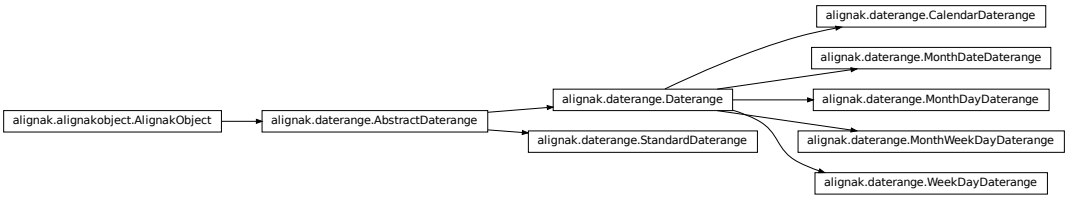


Graph GenericInterface :

Graph ActionBase :



Graph AbstractDaterange :


alignak.alignakobject.AlignakObject
alignak.daterange.AbstractDaterange
alignak.daterange.Daterange
alignak.daterange.StandardDaterange
alignak.daterange.CalendarDaterange
alignak.daterange.MonthDateDaterange
alignak.daterange.MonthDayDaterange
alignak.daterange.MonthWeekDayDaterange
alignak.daterange.WeekDayDaterange

CHAPTER 2

alignak package

2.1 Subpackages

2.1.1 alignak.bin package

Submodules

alignak.bin.alignak_arbiter module

This is the class of the Arbiter. Its role is to read configuration, cut it, and send it to other elements like schedulers, reactionners or pollers. It is also responsible for the high availability feature. For example, if a scheduler dies, it sends the late scheduler's conf to another scheduler available.

`alignak.bin.alignak_arbiter.`**`main`**`()`
Parse args and run main daemon function

Returns None

alignak.bin.alignak_broker module

This class is an interface for the Broker The broker listens to the Arbiter for the configuration sent through the given port as first argument. The configuration sent by the arbiter specifies from which schedulers the broker will take broks. When the broker is already launched and has its own conf, it keeps on listening the arbiter (one a timeout) In case the arbiter has a new conf to send, the broker forget its old schedulers (and their associated broks) and take the new ones instead.

`alignak.bin.alignak_broker.`**`main`**`()`
Parse args and run main daemon function

Returns None

alignak.bin.alignak_environment module

alignak-environment command line interface:

```
Usage:
    alignak [-h|--help]
    alignak [-v|--verbose] <cfg_file>

Options:
    -h, --help          Show this usage screen.
    -v, --verbose       Run in verbose mode (print information on the console output)

Output:
    This script will parse the provided configuration file and it will output all the
    variables defined in this file as Linux/Unix shell export variables.

    As an example for a file as the default ./etc/alignak.ini, the script will output:
        export ALIGNAK_CONFIGURATION_DIST=/usr/local/
        export ALIGNAK_CONFIGURATION_DIST_BIN=/usr/local//bin
        export ALIGNAK_CONFIGURATION_DIST_ETC=/usr/local//etc/alignak
        export ALIGNAK_CONFIGURATION_DIST_VAR=/usr/local//var/lib/alignak
        export ALIGNAK_CONFIGURATION_DIST_RUN=/usr/local//var/run/alignak
        export ALIGNAK_CONFIGURATION_DIST_LOG=/usr/local//var/log/alignak
        export ALIGNAK_CONFIGURATION_CONFIG_NAME='Alignak global configuration'
        export ALIGNAK_CONFIGURATION_ALIGNAK_NAME='My Alignak'
        export ALIGNAK_CONFIGURATION_USER=alignak
        export ALIGNAK_CONFIGURATION_GROUP=alignak
        ...
        export DAEMON_ARBITER_MASTER_DIST=/usr/local/
        export DAEMON_ARBITER_MASTER_DIST_BIN=/usr/local//bin
        export DAEMON_ARBITER_MASTER_DIST_ETC=/usr/local//etc/alignak
        export DAEMON_ARBITER_MASTER_DIST_VAR=/usr/local//var/lib/alignak
        export DAEMON_ARBITER_MASTER_DIST_RUN=/usr/local//var/run/alignak
        export DAEMON_ARBITER_MASTER_DIST_LOG=/usr/local//var/log/alignak
        export DAEMON_ARBITER_MASTER_CONFIG_NAME='Alignak global configuration'
        export DAEMON_ARBITER_MASTER_ALIGNAK_NAME='My Alignak'
        export DAEMON_ARBITER_MASTER_USER=alignak
        export DAEMON_ARBITER_MASTER_GROUP=alignak
        ...
        export DAEMON_SCHEDULER_MASTER_DIST=/usr/local/
        export DAEMON_SCHEDULER_MASTER_DIST_BIN=/usr/local//bin
        ...
        export ALIGNAK_VERSION=1.0.0

    The export directives consider that shell variables must only contain [A-Za-z0-9_]
    in their name. All non alphanumeric characters are replaced with an underscore.
    The value of the variables is quoted to be shell-valid: escaped quotes, empty␣
↪strings,...

    NOTE: this script manages the full Ini file format used by the Python␣
↪ConfigParser:
    default section, variables interpolation

    NOTE: this script also adds the current Alignak version to the content of the
    configuration file

Use cases:
    Displays this usage screen
```

(continues on next page)

(continued from previous page)

```
        alignak-environment (-h | --help)

    Parse Alignak configuration file and define environment variables
        cfg_file ../etc/alignak-realm2.ini

    Parse Alignak configuration file and define environment variables and print␣
↪information
        cfg_file -v ../etc/alignak-realm2.ini

    Exit code:
        0 if required operation succeeded
        1 if the required file does not exist
        2 if the required file is not correctly formatted
        3 if interpolation variables are not correctly declared/used in the␣
↪configuration file

        64 if command line parameters are not used correctly
```

class alignak.bin.alignak_environment.**AlignakConfigParser**(*args=None*)

Bases: object

Class to parse the Alignak main configuration file

get_alignak_configuration(*section='alignak-configuration'*, *legacy_cfg=False*, *macros=False*)

Get the Alignak configuration parameters. All the variables included in the SECTION_CONFIGURATION section except the variables starting with 'cfg' and the macros.

If *lecagy_cfg* is True, this function only returns the variables included in the SECTION_CONFIGURATION section except the variables starting with 'cfg'

If *macros* is True, this function only returns the variables included in the SECTION_CONFIGURATION section that are considered as macros

Parameters

- **section** (*str*) – name of the sectio nto search for
- **legacy_cfg** (*bool*) – only get the legacy cfg declarations
- **macros** (*bool*) – only get the macros declarations

Returns a dict containing the Alignak configuration parameters

get_alignak_macros()

Get the Alignak macros.

Returns a dict containing the Alignak macros

get_daemons(*daemon_name=None*, *daemon_type=None*)

Get the daemons configuration parameters

If name is provided, get the configuration for this daemon, else, If type is provided, get the configuration for all the daemons of this type, else get the configuration of all the daemons.

Parameters

- **daemon_name** – the searched daemon name
- **daemon_type** – the searched daemon type

Returns a dict containing the daemon(s) configuration parameters

get_defaults()
Get all the parameters defined in the DEFAULT ini file section...

Returns a dict containing the default parameters

get_legacy_cfg_files()
Get the Alignak monitored configuration files.

Returns a dict containing the Alignak legacy configuration files

get_modules(*name=None*, *daemon_name=None*, *names_only=True*)
Get the modules configuration parameters

If name is provided, get the configuration for this module, else, If daemon_name is provided, get the configuration for all the modules of this daemon, else get the configuration of all the modules.

Parameters

- **name** – the searched module name
- **daemon_name** – the modules of this daemon
- **names_only** – if True only returns the modules names, else all the configuration data

Returns a dict containing the module(s) configuration parameters

parse()
Check if some extra configuration files are existing in an *alignak.d* sub directory near the found configuration file.

Parse the Alignak configuration file(s)

Exit the script if some errors are encountered.

Returns True/False

write(*env_file*)
Write the Alignak configuration to a file

Parameters **env_file** (*str*) – file name to dump the configuration

Returns True/False

alignak.bin.alignak_environment.**main**()
Main function

alignak.bin.alignak_poller module

This class is the application that launches checks The poller listens to the Arbiter for the configuration sent through the given port as first argument. The configuration sent by the arbiter specifies from which schedulers the poller will take its checks. When the poller is already launched and has its own conf, it keeps on listening the arbiter In case the arbiter has a new conf to send, the poller forget its old schedulers (and the associated checks) and take the new ones instead.

alignak.bin.alignak_poller.**main**()
Parse args and run main daemon function

Returns None

alignak.bin.alignak_reactionner module

This class is an application that launches actions like notifications or event handlers The reactionner listens to the Arbiter for the configuration sent through the given port as first argument. The configuration sent by the arbiter specifies from which schedulers the will take actions. When the reactionner is already launched and has its own conf, it keeps on listening the arbiter In case the arbiter has a new conf to send, the reactionner forget its old schedulers (and the associated actions) and take the new ones instead.

alignak.bin.alignak_reactionner.**main**()
Parse args and run main daemon function

Returns None

alignak.bin.alignak_receiver module

This class is an interface for the Receiver The receiver listens to the Arbiter for the configuration sent through the given port as first argument. The configuration sent by the arbiter specifies from which schedulers the receiver will take broks. When the receiver is already launched and has its own conf, it keeps on listening the arbiter In case the arbiter has a new conf to send, the receiver forget its old schedulers (and their associated broks) and take the new ones instead.

alignak.bin.alignak_receiver.**main**()
Parse args and run main daemon function

Returns None

alignak.bin.alignak_scheduler module

This class is the application in charge of scheduling The scheduler listens to the Arbiter for the configuration sent through the given port as first argument. The configuration sent by the arbiter specifies which checks and actions the scheduler must schedule, and a list of reactionners and pollers to execute them When the scheduler is already launched and has its own conf, it keeps on listening the arbiter In case the arbiter has a new conf to send, the scheduler is stopped and a new one is created.

alignak.bin.alignak_scheduler.**main**()
Parse args and run main daemon function

Returns None

Module contents

This file is to be imported by every Alignak service component: Arbiter, Scheduler, etc. It just checks for the main requirement of Alignak.

2.1.2 alignak.daemons package

Submodules

alignak.daemons.arbiterdaemon module

This module provide Arbiter class used to run a arbiter daemon

class alignak.daemons.arbiterdaemon.**Arbiter**(***kwargs*)
Bases: *alignak.daemon.Daemon*

Arbiter class. Referenced as "app" in most Interface

Class to manage the Arbiter daemon. The Arbiter is the one that rules them all...

add(*elt*)
Generic function to add objects to the daemon internal lists. Manage Broks, External commands

Parameters **elt** (*alignak.AlignakObject*) – objects to add

Returns None

check_and_log_tp_activation_change()
Raise log for timeperiod change (useful for debug)

Returns None

configuration_dispatch(*not_configured=None*)
Monitored configuration preparation and dispatch

Returns None

daemons_check()
Manage the list of Alignak launched daemons

Check if the daemon process is running

Returns True if all daemons are running, else False

daemons_reachability_check()
Manage the list of Alignak launched daemons

Check if the daemon process is running

Returns True if all daemons are running, else False

daemons_start(*run_daemons=True*)
Manage the list of the daemons in the configuration

Check if the daemon needs to be started by the Arbiter.

If so, starts the daemon if *run_daemons* is True

Parameters **run_daemons** (*bool*) – run the daemons or make a simple check

Returns True if all daemons are running, else False. always True for a simple check

daemons_stop(*timeout=30*, *kill_children=False*)
Stop the Alignak daemons

Iterate over the self-launched daemons and their children list to send a TERM Wait for daemons to terminate and then send a KILL for those that are not yet stopped

As a default behavior, only the launched daemons are killed, not their children. Each daemon will manage its children killing

Parameters

- **timeout** (*int*) – delay to wait before killing a daemon
- **kill_children** (*bool*) – also kill the children (defaults to False)

Returns True if all daemons stopped

do_before_loop()
Called before the main daemon loop.

Returns None

do_loop_turn()
Loop turn for Arbiter

If not a master daemon, wait for my master death... Else, run: * Check satellites are alive * Check and dispatch (if needed) the configuration * Get broks and external commands from the satellites * Push broks and external commands to the satellites

Returns None

get_alignak_status(*details=False*)
Push the alignak overall state as a passive check

Build all the daemons overall state as a passive check that can be notified to the Alignak WS

The Alignak Arbiter is considered as an host which services are all the Alignak running daemons. An Alignak daemon is considered as a service of an Alignak host.

As such, it reports its status as a passive service check formatted as defined for the Alignak WS module (see http://alignak-module-ws.readthedocs.io)

Returns A dict with the following structure

:: {

'name': 'type and name of the daemon', 'livestate': {

'state': "ok", 'output': "state message", 'long_output': "state message - longer ... if any", 'perf_data': "daemon metrics (if any...)"

} "services": {

"daemon-1": { 'name': 'type and name of the daemon', 'livestate': {

'state': "ok", 'output': "state message", 'long_output': "state message - longer ... if any", 'perf_data': "daemon metrics (if any...)"

}

} .../... "daemon-N": {

'name': 'type and name of the daemon', 'livestate': {

'state': "ok", 'output': "state message", 'long_output': "state message - longer ... if any", 'perf_data': "daemon metrics (if any...)"

}

}

}

}

Return type dict

get_broks_from_satellites()
Get broks from my all internal satellite links

The arbiter get the broks from ALL the known satellites

Returns None

get_daemon_stats(*details=False*)
Increase the stats provided by the Daemon base class

Returns stats dictionary

Return type dict

get_external_commands()
Get the external commands

Returns External commands list

Return type list

get_initial_broks_from_satellites()
Get initial broks from my internal satellite links

Returns None

get_livesynthesis()
Get the schedulers satellites live synthesis

Returns compiled livesynthesis dictionary

Return type dict

get_managed_configurations()
Get the configuration managed by this arbiter

This is used by the master arbiter to get information from its spare arbiter

Returns a dict of arbiter links (only one) with instance_id as key and

hash, push_flavor and configuration identifier as values :rtype: dict

get_monitoring_problems()
Get the schedulers satellites problems list

Returns problems dictionary

Return type dict

load_modules_alignak_configuration()
Load Alignak configuration from the arbiter modules If module implements get_alignak_configuration, call this function

Parameters **raw_objects** (*dict*) – raw objects we got from reading config files

Returns None

load_modules_configuration_objects(*raw_objects*)
Load configuration objects from arbiter modules If module implements get_objects arbiter will call it and add create objects

Parameters **raw_objects** (*dict*) – raw objects we got from reading config files

Returns None

load_monitoring_config_file(*clean=True*)
Load main configuration file (alignak.cfg):

```
* Read all files given in the -c parameters
* Read all .cfg files in cfg_dir
* Read all files in cfg_file
* Create objects (Arbiter, Module)
* Set HTTP links info (ssl etc)
```

(continues on next page)

(continued from previous page)

```
* Load its own modules
* Execute read_configuration hook (for arbiter modules)
* Create all objects (Service, Host, Realms ...)
* "Compile" configuration (Linkify, explode, apply inheritance, fill default␣
↪values ...)
* Cut conf into parts and prepare it for sending
```

The clean parameter is useful to load a configuration without removing the properties only used to parse the configuration and create the objects. Some utilities (like alignak-backend-import script) may need to avoid the cleaning ;)

Parameters **clean** (*bool*) – set True to clean the created items

Returns None

main()

Main arbiter function:

```
* Set logger
* Init daemon
* Launch modules
* Endless main process loop
```

Returns None

manage_signal(*sig*, *frame*)

Manage signals caught by the process Specific behavior for the arbiter when it receives a sigkill or sigterm

Parameters

- **sig** (*str*) – signal caught by the process
- **frame** – current stack frame

Returns None

properties = {'active': <Property <class 'alignak.property.BoolProp'>, default: True

push_broks_to_broker()

Send all broks from arbiter internal list to broker

The arbiter get some broks and then pushes them to all the brokers.

Returns None

push_external_commands_to_schedulers()

Send external commands to schedulers

Returns None

request_stop(*message=''*, *exit_code=0*)

Stop the Arbiter daemon

Returns None

setup_new_conf()

Setup a new configuration received from a Master arbiter.

TODO: perharps we should not accept the configuration or raise an error if we do not find our own configuration data in the data. Thus this should never happen... :return: None

start_daemon(*satellite*)
Manage the list of detected missing daemons

If the daemon does not in exist *my_daemons*, then:

- prepare daemon start arguments (port, name and log file)
- start the daemon
- make sure it started correctly

Parameters **satellite** (SatelliteLink) – the satellite for which a daemon is to be started

Returns True if the daemon started correctly

wait_for_master_death()
Wait for a master timeout and take the lead if necessary

Returns None

alignak.daemons.brokerdaemon module

This module provide Broker class used to run a broker daemon

class alignak.daemons.brokerdaemon.**Broker**(***kwargs*)
Bases: *alignak.satellite.BaseSatellite*

Class to manage a Broker daemon A Broker is used to get data from Scheduler and send them to modules. These modules in most cases export to other software, databases...

add(*elt*)
Generic function to add objects to the daemon internal lists. Manage Broks, External commands and Messages (from modules queues)

Parameters **elt** (*alignak.AlignakObject*) – object to add

Returns None

clean_previous_run()
Clean all (when we received new conf)

Returns None

do_loop_turn()
Loop used to: * get initial status broks * check if modules are alive, if not restart them * get broks from ourself, the arbiters and our satellites * add broks to the queue of each external module * manage broks with each internal module

If the internal broks management is longer than 0.8 seconds, postpone to hte next loop turn to avoid overloading the broker daemon.

Returns None

get_arbiter_broks()
Get the broks from the arbiters, but as the arbiter_broks list can be push by arbiter without Global lock, we must protect this with a lock

TODO: really? check this arbiter behavior!

Returns None

get_daemon_stats(*details=False*)
Increase the stats provided by the Daemon base class

Returns stats dictionary

Return type dict

get_internal_broks()
Get all broks from self.broks_internal_raised and append them to our broks to manage

Returns None

get_new_broks()
Get new broks from our satellites

Returns None

main()
Main function, will loop forever

Returns None

manage_brok(*brok*)
Get a brok. We put brok data to the modules

Parameters **brok** (*object*) – object with data

Returns None

properties = {'active': <Property <class 'alignak.property.BoolProp'>, default: True

setup_new_conf()
Broker custom setup_new_conf method

This function calls the base satellite treatment and manages the configuration needed for a broker daemon:
- get and configure its pollers, reactionners and receivers relation - configure the modules

Returns None

alignak.daemons.pollerdaemon module

This modules provides class for the Poller daemon

class alignak.daemons.pollerdaemon.**Poller**(***kwargs*)
Bases: *alignak.satellite.Satellite*

Poller class. Referenced as "app" in most Interface

do_actions = False

do_checks = True

my_type = 'poller'

properties = {'active': <Property <class 'alignak.property.BoolProp'>, default: True

alignak.daemons.reactionnerdaemon module

This module provide Reactionner class used to launch notifications or event handlers

class alignak.daemons.reactionnerdaemon.**Reactionner**(***kwargs*)
Bases: *alignak.satellite.Satellite*

This class is an application that launches actions for the schedulers Actions can be:

Notifications Event handlers

When running the Reactionner will : Respond to pings from Arbiter Listen for new configurations from Arbiter

The configuration consists of a list of Schedulers for which the Reactionner will launch actions for.

do_actions = True

do_checks = False

my_type = 'reactionner'

properties = {'active': <Property <class 'alignak.property.BoolProp'>, default: True

alignak.daemons.receiverdaemon module

This module provide Receiver class used to run a receiver daemon

class alignak.daemons.receiverdaemon.**Receiver**(***kwargs*)
Bases: *alignak.satellite.Satellite*

Receiver class. Referenced as "app" in most Interface

add(*elt*)
Generic function to add objects to the daemon internal lists. Manage Broks, External commands

Parameters **elt** (*alignak.AlignakObject*) – object to add

Returns None

do_loop_turn()
Receiver daemon main loop

Returns None

get_daemon_stats(*details=False*)
Increase the stats provided by the Daemon base class

Returns stats dictionary

Return type dict

get_external_commands_from_arbiters()
Get external commands from our arbiters

As of now, only the arbiter are requested to provide their external commands that the receiver will push to all the known schedulers to make them being executed.

Returns None

main()
Main receiver function Init daemon and loop forever

Returns None

my_type = 'receiver'

properties = {'active': <Property <class 'alignak.property.BoolProp'>, default: True

push_external_commands_to_schedulers()
Push received external commands to the schedulers

Returns None

setup_new_conf()
Receiver custom setup_new_conf method

This function calls the base satellite treatment and manages the configuration needed for a receiver daemon: - get and configure its satellites - configure the modules

Returns None

alignak.daemons.schedulerdaemon module

This module provide Alignak which is the main scheduling daemon class

class alignak.daemons.schedulerdaemon.**Alignak**(***kwargs*)
Bases: *alignak.satellite.BaseSatellite*

Scheduler class. Referenced as "app" in most Interface

clean_previous_run()
Clean variables from previous configuration

Returns None

compensate_system_time_change(*difference*)
Compensate a system time change of difference for all hosts/services/checks/notifs

Parameters **difference** (*int*) – difference in seconds

Returns None

do_before_loop()
Stop the scheduling process

do_loop_turn()
Scheduler loop turn

Simply run the Alignak scheduler loop

This is called when a configuration got received by the scheduler daemon. As of it, check if the first scheduling has been done... and manage this.

Returns None

get_broks(*broker_name*)
Send broks to a specific broker

Parameters **broker_name** (*str*) – broker name to send broks

Greturn dict of brok for this broker

Return type dict[*alignak.brok.Brok*]

get_daemon_stats(*details=False*)
Increase the stats provided by the Daemon base class

Returns stats dictionary

Return type dict

get_managed_configurations()
Get the configurations managed by this scheduler

The configuration managed by a scheduler is the self configuration got by the scheduler during the dispatching.

Returns a dict of scheduler links with instance_id as key and

hash, push_flavor and configuration identifier as values :rtype: dict

get_monitoring_problems()
Get the current scheduler livesynthesis

Returns live synthesis and problems dictionary

Return type dict

main()
Main function for Scheduler, launch after the init:

```
* Init daemon
* Load module manager
* Launch main loop
* Catch any Exception that occurs
```

Returns None

properties = {'active': <Property <class 'alignak.property.BoolProp'>, default: True

setup_new_conf()
Setup new conf received for scheduler

Returns None

Module contents

Daemon package init, nothing necessary to do here

2.1.3 alignak.http package

Submodules

alignak.http.arbiter_interface module

This module provides a specific HTTP interface for a Arbiter.

class alignak.http.arbiter_interface.**ArbiterInterface**(*app*)
Bases: *alignak.http.generic_interface.GenericInterface*

This module provide a specific HTTP interface for an Arbiter daemon.

backend_notification(*event=None*, *parameters=None*)
Possible events are: - creation, for a realm or an host creation - deletion, for a realm or an host deletion

Calls the reload configuration function if event is creation or deletion

Else, nothing for the moment!

In case of any error, this function returns an object containing some properties: '_status': 'ERR' because of the error *_message*: some more explanations about the error

The *_status* field is 'OK' with an according *_message* to explain what the Arbiter will do depending upon the notification.

Returns dict

command(*command=None*, *timestamp=None*, *element=None*, *host=None*, *service=None*, *user=None*, *parameters=None*)

Request to execute an external command

Allowed parameters are: *command*: mandatory parameter containing the whole command line or only the command name

timestamp: optional parameter containing the timestamp. If not present, the current timestamp is added in the command line

element: the targeted element that will be appended after the command name (*command*). If element contains a '/' character it is split to make an host and service.

host, *service* or *user*: the targeted host, service or user. Takes precedence over the *element* to target a specific element

parameters: the parameter that will be appended after all the arguments

When using this endpoint with the HTTP GET method, the semi colons that are commonly used to separate the parameters must be replace with %3B! This because the ; is an accepted URL query parameters separator...

Indeed, the recommended way of using this endpoint is to use the HTTP POST method.

In case of any error, this function returns an object containing some properties: '_status': 'ERR' because of the error *_message*: some more explanations about the error

The *_status* field is 'OK' with an according *_message* to explain what the Arbiter will do depending upon the notification. The *command* property contains the formatted external command.

Returns dict

dump(*o_name=None*, *details=False*, *raw=False*)

Dump an host (all hosts) from the arbiter.

The arbiter will get the host (all hosts) information from all its schedulers.

This gets the main host information from the scheduler. If details is set, then some more information are provided. This will not get all the host known attributes but only a reduced set that will inform about the host and its services status

If raw is set the information are provided in two string lists formated as CSV strings. The first list element contains the hosts information and the second one contains the services information.

If an host name is provided, this function will get only this host information, else all the scheduler hosts are returned.

As an example (in raw format): {

scheduler-master-3: [

["type;host;name;last_check;state_id;state;state_type;is_problem; is_impact;output", "localhost;host;localhost;1532451740;0;UP;HARD;False;False; Host assumed to be UP", "host_2;host;host_2;1532451988;1;DOWN;HARD;True;False;I am always Down"

], [

"type;host;name", "host_2;service;dummy_no_output;1532451981;0;OK;HARD;False;True;
Service internal check result: 0", "host_2;service;dummy_warning;1532451960;4;UNREACHABLE;HARD;Fals
host_2-dummy_warning-1", "host_2;service;dummy_unreachable;1532451987;4;UNREACHABLE;HARD;Fals
host_2-dummy_unreachable-4", "host_2;service;dummy_random;1532451949;4;UNREACHABLE;HARD;Fals
Service internal check result: 2", "host_2;service;dummy_ok;1532452002;0;OK;HARD;False;True;host_2",
"host_2;service;dummy_critical;1532451953;4;UNREACHABLE;HARD;False;True;

host_2-dummy_critical-2", "host_2;service;dummy_unknown;1532451945;4;UNREACHABLE;HARD;False;T
host_2-dummy_unknown-3", "host_2;service;dummy_echo;1532451973;4;UNREACHABLE;HARD;False;Tru

]

], scheduler-master-2: [[

"type;host;name;last_check;state_id;state;state_type;is_problem;is_impact;output", "host_0;host;host_0;1532451993;0;UP;HARD;False;False;I am always Up", "BR_host;host;BR_host;1532451991;0;UP;HARD;False;False;Host assumed to be UP"

], [

"type;host;name;last_check;state_id;state;state_type;is_problem;is_impact;output", "host_0;service;dummy_no_output;1532451970;0;OK;HARD;False;False; Service internal check result: 0", "host_0;service;dummy_unknown;1532451964;3;UNKNOWN;HARD;True;False; host_0-dummy_unknown-3", "host_0;service;dummy_random;1532451991;1;WARNING;HARD;True;False; Service internal check result: 1", "host_0;service;dummy_warning;1532451945;1;WARNING;HARD;True;False; host_0-dummy_warning-1", "host_0;service;dummy_unreachable;1532451986;4;UNREACHABLE;HARD;True;F host_0-dummy_unreachable-4", "host_0;service;dummy_ok;1532452012;0;OK;HARD;False;False;host_0", "host_0;service;dummy_critical;1532451987;2;CRITICAL;HARD;True;False; host_0-dummy_critical-2", "host_0;service;dummy_echo;1532451963;0;OK;HARD;False;False;", "BR_host;service;dummy_critical;1532451970;2;CRITICAL;HARD;True;False; BR_host-dummy_critical-2", "BR_host;service;BR_Simple_And;1532451895;1;WARNING;HARD;True;True;", "BR_host;service;dummy_unreachable;1532451981;4;UNREACHABLE;HARD;True;False; BR_host-dummy_unreachable-4", "BR_host;service;dummy_no_output;1532451975;0;OK;HARD;False;False; Service internal check result: 0", "BR_host;service;dummy_unknown;1532451955;3;UNKNOWN;HARD;True;Fal BR_host-dummy_unknown-3", "BR_host;service;dummy_echo;1532451981;0;OK;HARD;False;False;", "BR_host;service;dummy_warning;1532451972;1;WARNING;HARD;True;False; BR_host-dummy_warning-1", "BR_host;service;dummy_random;1532451976;4;UNREACHABLE;HARD;True;F Service internal check result: 4", "BR_host;service;dummy_ok;1532451972;0;OK;HARD;False;False;BR_host"

]

More information are available in the scheduler correponding API endpoint.

Parameters

- **o_type** (*str*) – searched object type
- **o_name** (*str*) – searched object name (or uuid)

Returns serialized object information

Return type str

events_log(*details=False*, *count=0*, *timestamp=0*)

Get the most recent Alignak events

If count is specifies it is the maximum number of events to return.

If timestamp is specified, events older than this timestamp will not be returned

The arbiter maintains a list of the most recent Alignak events. This endpoint provides this list.

The default format is: [

"2018-07-23 15:14:43 - E - SERVICE NOTIFICATION: guest;host_0;dummy_random;CRITICAL;1; notify-service-by-log;Service internal check result: 2", "2018-07-23 15:14:43 - E - SERVICE NOTIFICATION: admin;host_0;dummy_random;CRITICAL;1; notify-service-by-log;Service internal check result: 2", "2018-07-23 15:14:42 - E - SERVICE ALERT: host_0;dummy_critical;CRITICAL;SOFT;1;

host_0-dummy_critical-2", "2018-07-23 15:14:42 - E - SERVICE ALERT: host_0;dummy_random;CRITICAL;HARD;2; Service internal check result: 2", "2018-07-23 15:14:42 - I - SERVICE ALERT: host_0;dummy_unknown;UNKNOWN;HARD;2; host_0-dummy_unknown-3"

]

If you request on this endpoint with the *details* parameter (whatever its value...), you will get a detailed JSON output: [

{ timestamp: 1535517701.1817362, date: "2018-07-23 15:16:35", message: "SERVICE ALERT: host_11;dummy_echo;UNREACHABLE;HARD;2;", level: "info"

}, {

timestamp: 1535517701.1817362, date: "2018-07-23 15:16:32", message: "SERVICE NOTIFICATION: guest;host_0;dummy_random;OK;0;

notify-service-by-log;Service internal check result: 0",

level: "info"

}, {

timestamp: 1535517701.1817362, date: "2018-07-23 15:16:32", message: "SERVICE NOTIFICATION: admin;host_0;dummy_random;OK;0;

notify-service-by-log;Service internal check result: 0",

level: "info"

}, {

timestamp: 1535517701.1817362, date: "2018-07-23 15:16:32", message: "SERVICE ALERT: host_0;dummy_random;OK;HARD;2;

Service internal check result: 0",

level: "info"

}, {

timestamp: 1535517701.1817362, date: "2018-07-23 15:16:19", message: "SERVICE ALERT: host_11;dummy_random;OK;HARD;2;

Service internal check result: 0",

level: "info"

}

]

In this example, only the 5 most recent events are provided whereas the default value is to provide the 100 last events. This default counter may be changed thanks to the `events_log_count` configuration variable or `ALIGNAK_EVENTS_LOG_COUNT` environment variable.

The date format may also be changed thanks to the `events_date_format` configuration variable.

Returns list of the most recent events

Return type list

external_commands()

Get the external commands from the daemon

Use a lock for this function to protect

Returns serialized external command list

Return type str

host()

Get a passive checks for an host and its services

This function builds the external commands corresponding to the host and services provided information

Parameters

- **host_name** – host name
- **data** – dictionary of the host properties to be modified

Returns command line

livesynthesis()

Get Alignak live synthesis

This will return an object containing the properties of the *identity*, plus a *livesynthesis* object which contains 2 properties for each known scheduler: - _freshness, which is the timestamp when the provided data were fetched - livesynthesis, which is an object with the scheduler live synthesis.

An *_overall* fake scheduler is also contained in the schedulers list to provide the cumulated live synthesis. Before sending the results, the arbiter sums-up all its schedulers live synthesis counters in the *_overall* live synthesis.

{ ...

"livesynthesis": {

"_overall": { "_freshness": 1528947526, "livesynthesis": {

"hosts_total": 11, "hosts_not_monitored": 0, "hosts_up_hard": 11, "hosts_up_soft": 0, "hosts_down_hard": 0, "hosts_down_soft": 0, "hosts_unreachable_hard": 0, "hosts_unreachable_soft": 0, "hosts_flapping": 0, "hosts_problems": 0, "hosts_acknowledged": 0, "hosts_in_downtime": 0, "services_total": 100, "services_not_monitored": 0, "services_ok_hard": 70, "services_ok_soft": 0, "services_warning_hard": 4, "services_warning_soft": 6, "services_critical_hard": 6, "services_critical_soft": 4, "services_unknown_hard": 3, "services_unknown_soft": 7, "services_unreachable_hard": 0, "services_unreachable_soft": 0, "services_flapping": 0, "services_problems": 0, "services_acknowledged": 0, "services_in_downtime": 0 }

}

}, "scheduler-master": {

"_freshness": 1528947522, "livesynthesis": {

"hosts_total": 11, "hosts_not_monitored": 0, "hosts_up_hard": 11, "hosts_up_soft": 0, "hosts_down_hard": 0, "hosts_down_soft": 0, "hosts_unreachable_hard": 0, "hosts_unreachable_soft": 0, "hosts_flapping": 0, "hosts_problems": 0, "hosts_acknowledged": 0, "hosts_in_downtime": 0, "services_total": 100, "services_not_monitored": 0, "services_ok_hard": 70, "services_ok_soft": 0, "services_warning_hard": 4, "services_warning_soft": 6, "services_critical_hard": 6, "services_critical_soft": 4, "services_unknown_hard": 3, "services_unknown_soft": 7, "services_unreachable_hard": 0, "services_unreachable_soft": 0, "services_flapping": 0, "services_problems": 0, "services_acknowledged": 0, "services_in_downtime": 0 }

}

}

}

}

Returns scheduler live synthesis

Return type dict

monitoring_problems()

Get Alignak detailed monitoring status

This will return an object containing the properties of the *identity*, plus a *problems* object which contains 2 properties for each known scheduler: - _freshness, which is the timestamp when the provided data were fetched - problems, which is an object with the scheduler known problems:

{ ...

"problems": {

"scheduler-master": { "_freshness": 1528903945, "problems": {

"fdfc986d-4ab4-4562-9d2f-4346832745e6": { "last_state": "CRITICAL", "service": "dummy_critical", "last_state_type": "SOFT", "last_state_update": 1528902442, "last_hard_state": "CRITICAL", "last_hard_state_change": 1528902442, "last_state_change": 1528902381, "state": "CRITICAL", "state_type": "HARD", "host": "host-all-8", "output": "Hi, checking host-all-8/dummy_critical -> exit=2"

}, "2445f2a3-2a3b-4b13-96ed-4cfb60790e7e": {

"last_state": "WARNING", "service": "dummy_warning", "last_state_type": "SOFT", "last_state_update": 1528902463, "last_hard_state": "WARNING", "last_hard_state_change": 1528902463, "last_state_change": 1528902400, "state": "WARNING", "state_type": "HARD", "host": "host-all-6", "output": "Hi, checking host-all-6/dummy_warning -> exit=1"

}

}

}

}

Returns schedulers live synthesis list

Return type dict

object(*o_type*, *o_name=None*)

Get a monitored object from the arbiter.

Indeed, the arbiter requires the object from its schedulers. It will iterate in its schedulers list until a matching object is found. Else it will return a Json structure containing _status and _message properties.

When found, the result is a serialized object which is a Json structure containing: - content: the serialized object content - __sys_python_module__: the python class of the returned object

The Alignak unserialize function of the alignak.misc.serialization package allows to restore the initial object.

```
from alignak.misc.serialization import unserialize
from alignak.objects.hostgroup import Hostgroup
raw_data = req.get("http://127.0.0.1:7768/object/hostgroup/allhosts")
print("Got: %s / %s" % (raw_data.status_code, raw_data.content))
assert raw_data.status_code == 200
```

(continues on next page)

(continued from previous page)

```
object = raw_data.json()
group = unserialize(object, True)
assert group.__class__ == Hostgroup
assert group.get_name() == 'allhosts'
```

As an example: {

"__sys_python_module__": "alignak.objects.hostgroup.Hostgroup", "content": {

"uuid": "32248642-97dd-4f39-aaa2-5120112a765d", "name": "", "hostgroup_name": "allhosts", "use": [], "tags": [], "alias": "All Hosts", "notes": "", "definition_order": 100, "register": true, "unknown_members": [], "notes_url": "", "action_url": "",

"imported_from": "unknown", "conf_is_correct": true, "configuration_errors": [], "configuration_warnings": [], "realm": "", "downtimes": {}, "hostgroup_members": [], "members": [

"553d47bc-27aa-426c-a664-49c4c0c4a249", "f88093ca-e61b-43ff-a41e-613f7ad2cea2", "df1e2e13-552d-43de-ad2a-fe80ad4ba979", "d3d667dd-f583-4668-9f44-22ef3dcb53ad"

]

}

}

Parameters

- **o_type** (*str*) – searched object type
- **o_name** (*str*) – searched object name (or uuid)

Returns serialized object information

Return type str

problems()

Alias for monitoring_problems

query()

Request object passed to datasource.query function:

{ 'timezone': 'browser', 'panelId': 38, 'range': {

'from': '2018-08-29T02:38:09.633Z', 'to': '2018-08-29T03:38:09.633Z', 'raw': {'from': 'now-1h', 'to': 'now'}

}, 'rangeRaw': {'from': 'now-1h', 'to': 'now'}, 'interval': '10s', 'intervalMs': 10000, 'targets': [

{ 'target': 'problems', 'refId': 'A', 'type': 'table'}

], 'format': 'json', 'maxDataPoints': 314, 'scopedVars': {

'__interval': {'text': '10s', 'value': '10s'}, '__interval_ms': {'text': 10000, 'value': 10000}

}

}

Only the first target is considered. If several targets are required, an error is raised.

The target is a string that is searched in the target_queries dictionary. If found the corresponding query is executed and the result is returned.

Table response from datasource.query. An array of:

[

{ “type”: “table”, “columns”: [

{ “text”: “Time”, “type”: “time”, “sort”: true, “desc”: true,

}, {

“text”: “mean”,

}, {

“text”: “sum”,

}

], “rows”: [

[1457425380000, null, null

], [

1457425370000, 1002.76215352, 1002.76215352

],

]

}

] :return: See upper comment :rtype: list

realms(*details=False*)

Return the realms / satellites configuration

Returns an object containing the hierarchical realms configuration with the main information about each realm: {

All: {

satellites: {

pollers: [“poller-master”

], reactionners: [

“reactionner-master”

], schedulers: [

“scheduler-master”, “scheduler-master-3”, “scheduler-master-2”

], brokers: [“broker-master”], receivers: [“receiver-master”, “receiver-nsca”]

}, children: { }, name: “All”, members: [

“host_1”, “host_0”, “host_3”, “host_2”, “host_11”, “localhost”

], level: 0

}, North: {

...

}

}

Sub realms defined inside a realm are provided in the *children* property of their parent realm and they contain the same information as their parent.. The *members* realm contain the list of the hosts members of the realm.

If `details` is required, each realm will contain more information about each satellite involved in the realm management: {

All: {

satellites: {

pollers: [

{ passive: false, name: "poller-master", livestate_output: "poller/poller-master is up and running.", reachable: true, uri: "http://127.0.0.1:7771/", alive: true, realm_name: "All", manage_sub_realms: true, spare: false, polling_interval: 5, configuration_sent: true, active: true, livestate: 0, max_check_attempts: 3, last_check: 1532242300.593074, type: "poller"

}

], reactionners: [

{ passive: false, name: "reactionner-master", livestate_output: "reactionner/reactionner-master is up and running.", reachable: true, uri: "http://127.0.0.1:7769/", alive: true, realm_name: "All", manage_sub_realms: true, spare: false, polling_interval: 5, configuration_sent: true, active: true, livestate: 0, max_check_attempts: 3, last_check: 1532242300.587762, type: "reactionner"

}

]

Returns dict containing realms / satellites

Return type dict

reload_configuration()

Ask to the arbiter to reload the monitored configuration

Note tha the arbiter will not reload its main configuration file (eg. alignak.ini) but it will reload the monitored objects from the Nagios legacy files or from the Alignak backend!

In case of any error, this function returns an object containing some properties: '_status': 'ERR' because of the error *_message*: some more explanations about the error

Returns True if configuration reload is accepted

satellites_configuration()

Return all the configuration data of satellites

Returns dict containing satellites data

Output looks like this

```
{'arbiter' : [{'property1':'value1' ..}, {'property2', 'value11' ..}, ..],
```

'scheduler': [..], 'poller': [..], 'reactionner': [..], 'receiver': [..],

'broker: [..]'

}

Return type dict

satellites_list(*daemon_type=''*)

Get the arbiter satellite names sorted by type

Returns a list of the satellites as in: {

reactionner: ["reactionner-master"

], broker: [

"broker-master"

], arbiter: [

"arbiter-master"

], scheduler: [

"scheduler-master-3", "scheduler-master", "scheduler-master-2"

], receiver: [

"receiver-nsca", "receiver-master"

], poller: [

"poller-master"

]

}

If a specific daemon type is requested, the list is reduced to this unique daemon type: {

scheduler: ["scheduler-master-3", "scheduler-master", "scheduler-master-2"

]

}

Parameters **daemon_type** (*str*) – daemon type to filter

Returns dict with key *daemon_type* and value list of daemon name

Return type dict

search()

Request available queries

Posted data: {u'target': u''}

Return the list of available target queries

Returns See upper comment

Return type list

status(*details=False*)

Get the overall alignak status

Returns a list of the satellites as in: {

services: [

{

livestate: { perf_data: "", timestamp: 1532106561, state: "ok", long_output: "", output: "all daemons are up and running."

}, name: "arbiter-master"

}, {

livestate: { name: "poller_poller-master", timestamp: 1532106561, long_output: "Realm: (True). Listening on: http://127.0.0.1:7771/", state: "ok", output: "daemon is alive and reachable.", perf_data: "last_check=1532106560.17"

}, name: "poller-master"

...

], variables: { }, livestate: {

timestamp: 1532106561, long_output: "broker-master - daemon is alive and reachable. poller-master - daemon is alive and reachable. reactionner-master - daemon is alive and reachable. receiver-master - daemon is alive and reachable. receiver-nsca - daemon is alive and reachable. scheduler-master - daemon is alive and reachable. scheduler-master-2 - daemon is alive and reachable. scheduler-master-3 - daemon is alive and reachable.", state: "up", output: "All my daemons are up and running.", perf_data: "

'servicesextinfo'=0 'businessimpactmodulations'=0 'hostgroups'=2 'resultmodulations'=0 'escalations'=0 'schedulers'=3 'hostsextinfo'=0 'contacts'=2 'servicedependencies'=0 'servicegroups'=1 'pollers'=1 'arbiters'=1 'receivers'=2 'macromodulations'=0 'reactionners'=1 'contactgroups'=2 'brokers'=1 'realms'=3 'services'=32 'commands'=11 'notificationways'=2 'timeperiods'=4 'modules'=0 'checkmodulations'=0 'hosts'=6 'hostdependencies'=0"

}, name: "My Alignak", template: {

notes: "", alias: "My Alignak", _templates: [

"alignak", "important"

], active_checks_enabled: false, passive_checks_enabled: true

}

}

Parameters **details** – Details are required (different from 0)

:type details bool

Returns dict with key *daemon_type* and value list of daemon name

Return type dict

system(*details=False*)

Return the realms / satellites configuration

Returns an object containing the hierarchical realms configuration with the main information about each realm: {

All: {

satellites: {

pollers: ["poller-master"

], reactionners: [

"reactionner-master"

], schedulers: [

"scheduler-master", "scheduler-master-3", "scheduler-master-2"

], brokers: ["broker-master"], receivers: ["receiver-master", "receiver-nsca"]

}, children: { }, name: "All", members: [

"host_1", "host_0", "host_3", "host_2", "host_11", "localhost"

], level: 0

}, North: {

...

}

}

Sub realms defined inside a realm are provided in the *children* property of their parent realm and they contain the same information as their parent.. The *members* realm contain the list of the hosts members of the realm.

If `details` is required, each realm will contain more information about each satellite involved in the realm management: {

All: {

satellites: {

pollers: [

{ passive: false, name: "poller-master", livestate_output: "poller/poller-master is up and running.", reachable: true, uri: "http://127.0.0.1:7771/", alive: true, realm_name: "All", manage_sub_realms: true, spare: false, polling_interval: 5, configuration_sent: true, active: true, livestate: 0, max_check_attempts: 3, last_check: 1532242300.593074, type: "poller"

}

], reactionners: [

{ passive: false, name: "reactionner-master", livestate_output: "reactionner/reactionner-master is up and running.", reachable: true, uri: "http://127.0.0.1:7769/", alive: true, realm_name: "All", manage_sub_realms: true, spare: false, polling_interval: 5, configuration_sent: true, active: true, livestate: 0, max_check_attempts: 3, last_check: 1532242300.587762, type: "reactionner"

}

]

Returns dict containing realms / satellites

Return type dict

alignak.http.broker_interface module

This module provide a specific HTTP interface for a Broker.

class alignak.http.broker_interface.**BrokerInterface**(*app*)
Bases: *alignak.http.generic_interface.GenericInterface*

This class provides specific HTTP functions for the Broker daemons.

alignak.http.cherrypy_extend module

This module provide extension functions for Cherrypy in order to parse specific HTTP content type See http://cherrypy.readthedocs.org/en/latest/pkg/cherrypy.html#module-cherrypy._cpreqbody for details about custom processors in Cherrypy

alignak.http.cherrypy_extend.**zlib_processor**(*entity*)
Read application/zlib data and put content into entity.params for later use.

Parameters **entity** (*cherrypy._cpreqbody.Entity*) – cherrypy entity

Returns None

alignak.http.client module

This module provides HTTPClient class. Used by daemon to connect to HTTP servers (other daemons)

class alignak.http.client.**HTTPClient**(*address=''*, *port=0*, *use_ssl=False*, *short_timeout=3*, *long_timeout=120*, *uri=''*, *strong_ssl=False*, *proxy=''*)
Bases: object

HTTPClient class use python request to communicate over HTTP Basically used to get / post to other daemons

get(*path*, *args=None*, *wait=False*)
GET an HTTP request to a daemon

Parameters

- **path** (*str*) – path to do the request
- **args** (*dict*) – args to add in the request
- **wait** (*bool*) – True for a long timeout

Returns None

make_timeout(*wait*)
Get short_timeout depending on wait time

Parameters **wait** (*bool*) – wait for a long timeout

Returns self.short_timeout if wait is short, self.long_timeout otherwise

Return type int

make_uri(*path*)
Create uri from path

Parameters **path** (*str*) – path to make uri

Returns self.uri + path

Return type str

post (*path*, *args*, *wait=False*)
POST an HTTP request to a daemon

Parameters

- **path** (*str*) – path to do the request
- **args** (*dict*) – args to add in the request
- **wait** (*bool*) – True for a long timeout

Returns Content of the HTTP response if server returned 200

Return type str

put (*path*, *args*, *wait=False*)
PUT and HTTP request to a daemon

Parameters

- **path** (*str*) – path to do the request
- **args** – data to send in the request

Returns Content of the HTTP response if server returned 200

Return type str

set_proxy (*proxy*)
Set HTTP proxy

Parameters **proxy** (*str*) – proxy url

Returns None

exception alignak.http.client.**HTTPClientConnectionException** (*uri*, *msg*)
Bases: exceptions.Exception

HTTP Connection Exception - raised when connection fails with the server. This specific exception is raised when a connection exception is catched.

Its attribute are: - uri: the requested URI, - msg: the exception message

exception alignak.http.client.**HTTPClientDataException** (*rsp_code*, *rsp_text*, *uri*)
Bases: exceptions.Exception

HTTP Data Exception - raised when the HTTP response is not OK (200)

Its attribute are: - rsp_code: the HTTP response code, - rsp_text: the HTTP response bodyeout.

exception alignak.http.client.**HTTPClientException**
Bases: exceptions.Exception

Simple HTTP Exception - raised for all requests exception except for a timeout

exception alignak.http.client.**HTTPClientTimeoutException** (*timeout*, *uri*)
Bases: exceptions.Exception

HTTP Timeout Exception - raised when no response issued by the server in the specified time frame. This specific exception is raised when a requests Timeout exception is catched.

Its attribute are: - uri: the requested URI, - timeout: the duration of the timeout.

alignak.http.daemon module

This module provide the HTTP daemon for Alignak inter daemon communication. It is mostly based on Cherrypy

class alignak.http.daemon.**HTTPDaemon**(*host*, *port*, *http_interface*, *use_ssl*, *ca_cert*, *ssl_key*, *ssl_cert*, *server_dh*, *thread_pool_size*, *log_file=None*, *icon_file=None*)

Bases: object

HTTP Server class. Mostly based on Cherrypy It uses CherryPyWSGIServer and daemon http_interface as Application

run()

Wrapper to start the CherryPy server

This function throws a PortNotFree exception if any socket error is raised.

Returns None

stop()

Wrapper to stop the CherryPy server

Returns None

exception alignak.http.daemon.**PortNotFree**

Bases: exceptions.Exception

Exception raised when port is already used by another application

alignak.http.generic_interface module

This module provide a generic HTTP interface for all satellites.

Any Alignak satellite have at least these functions exposed over network See : http://cherrypy.readthedocs.org/en/latest/tutorials.html for Cherrypy basic HTTP apps.

All the _ prefixed functions are for internal use only and they will not be documented in the /api endpoint.

class alignak.http.generic_interface.**GenericInterface**(*app*)

Bases: object

Interface for inter satellites communications

api()

List the methods available on the daemon Web service interface

Returns a list of methods and parameters

Return type dict

get_log_level()

Get the current daemon log level

Returns an object with the daemon identity and a *log_level* property.

running_id :return: current log level :rtype: str

identity()

Get the daemon identity

This will return an object containing some properties: - alignak: the Alignak instance name - version: the Alignak version - type: the daemon type - name: the daemon name

Returns daemon identity

Return type dict

index()
Wrapper to call api from /

This will return the daemon identity and main information

Returns function list

managed_configurations()
Get the arbiter configuration managed by the daemon

For an arbiter daemon, it returns an empty object

For all other daemons it returns a dictionary formated list of the scheduler links managed by the daemon:
{

'instance_id': { 'hash': , 'push_flavor': , 'managed_conf_id':

}

}

If a daemon returns an empty list, it means that it has not yet received its configuration from the arbiter.

Returns managed configuration

Return type list

set_log_level(*log_level=None*)
Set the current log level for the daemon

The *log_level* parameter must be in [DEBUG, INFO, WARNING, ERROR, CRITICAL]

In case of any error, this function returns an object containing some properties: '_status': 'ERR' because of the error *_message*: some more explanations about the error

Else, this function returns True

Parameters **log_level** (*str*) – a value in one of the above

Returns see above

Return type dict

stats(*details=False*)
Get statistics and information from the daemon

Returns an object with the daemon identity, the daemon start_time and some extra properties depending upon the daemon type.

All daemons provide these ones: - program_start: the Alignak start timestamp - spare: to indicate if the daemon is a spare one - load: the daemon load - modules: the daemon modules information - counters: the specific daemon counters

Parameters **details** – Details are required (different from 0)

:type details str

Returns daemon stats

Return type dict

stop_request(*stop_now='0'*)
Request the daemon to stop

If *stop_now* is set to '1' the daemon will stop now. Else, the daemon will enter the stop wait mode. In this mode the daemon stops its activity and waits until it receives a new *stop_now* request to stop really.

Parameters **stop_now** (*bool*) – stop now or go to stop wait mode

Returns None

alignak.http.scheduler_interface module

This module provide a specific HTTP interface for a Scheduler.

class alignak.http.scheduler_interface.**SchedulerInterface**(*app*)

Bases: *alignak.http.generic_interface.GenericInterface*

This module provide a specific HTTP interface for a Scheduler daemon.

dump(*o_name=None*, *details=False*, *raw=False*)

Dump an host (all hosts) from the scheduler.

This gets the main host information from the scheduler. If details is set, then some more information are provided. This will not get all the host known attributes but only a reduced set that will inform about the host and its services status

If raw is set the information are provided in two string lists formated as CSV strings. The first list element contains the hosts information and the second one contains the services information.

If an host name is provided, this function will get only this host information, else all the scheduler hosts are returned.

As an example (raw format): [

[# Host information “type;host;name;last_check;state_id;state;state_type;is_problem;is_impact;output”, “BR_host;host;BR_host;1532451511;0;UP;HARD;False;False;Host assumed to be UP”

], [# Services information

“type;host;name;last_check;state_id;state;state_type;is_problem;is_impact;output”, “BR_host;service;dummy_critical;1532451490;2;CRITICAL;SOFT;False;False; BR_host-dummy_critical-2”, “BR_host;service;BR_Simple_And;0;0;OK;HARD;False;False;”, “BR_host;service;dummy_unreachable;1532451501;4;UNREACHABLE;SOFT;False;False; BR_host-dummy_unreachable-4”, “BR_host;service;dummy_no_output;1532451495;0;OK;HARD;False;False; Service internal check result: 0”, “BR_host;service;dummy_unknown;1532451475;3;UNKNOWN;SOFT;False;F BR_host-dummy_unknown-3”, “BR_host;service;dummy_echo;1532451501;0;OK;HARD;False;False;”, “BR_host;service;dummy_warning;1532451492;1;WARNING;SOFT;False;False; BR_host-dummy_warning-1”, “BR_host;service;dummy_random;1532451496;2;CRITICAL;SOFT;False;False; Service internal check result: 2”, “BR_host;service;dummy_ok;1532451492;0;OK;HARD;False;False;BR_host”

]

]

As an example (json format): {

is_impact: false, name: “BR_host”, state: “UP”, last_check: 1532451811, state_type: “HARD”, host: “BR_host”, output: “Host assumed to be UP”, services: [

{ is_impact: false, name: “dummy_critical”, state: “CRITICAL”, last_check: 1532451790, state_type: “HARD”, host: “BR_host”, output: “BR_host-dummy_critical-2”, state_id: 2, type: “service”, is_problem: true

}, {

is_impact: true, name: “BR_Simple_And”, state: “WARNING”, last_check: 1532451775, state_type: “SOFT”, host: “BR_host”, output: “”, state_id: 1, type: “service”, is_problem: false

}, state_id: 0, type: “host”, is_problem: false

}

Parameters

- **o_name** (*str*) – searched host name (or uuid)
- **details** (*bool*) – less or more details
- **raw** (*bool*) – json or raw text format

Returns list of host and services information

Return type list

monitoring_problems()

Get Alignak scheduler monitoring status

Returns an object with the scheduler livesynthesis and the known problems

Returns scheduler live synthesis

Return type dict

object(*o_type*, *o_name=None*)

Get an object from the scheduler.

The result is a serialized object which is a Json structure containing: - content: the serialized object content - __sys_python_module__: the python class of the returned object

The Alignak unserialize function of the alignak.misc.serialization package allows to restore the initial object.

```
from alignak.misc.serialization import unserialize
from alignak.objects.hostgroup import Hostgroup
raw_data = req.get("http://127.0.0.1:7768/object/hostgroup/allhosts")
print("Got: %s / %s" % (raw_data.status_code, raw_data.content))
assert raw_data.status_code == 200
object = raw_data.json()
group = unserialize(object, True)
assert group.__class__ == Hostgroup
assert group.get_name() == 'allhosts'
```

As an example: {

“__sys_python_module__”: “alignak.objects.hostgroup.Hostgroup”, “content”: {

“uuid”: “32248642-97dd-4f39-aaa2-5120112a765d”, “name”: “”, “hostgroup_name”: “allhosts”, “use”: [], “tags”: [], “alias”: “All Hosts”, “notes”: “”, “definition_order”: 100, “register”: true, “unknown_members”: [], “notes_url”: “”, “action_url”: “”,

“imported_from”: “unknown”, “conf_is_correct”: true, “configuration_errors”: [], “configuration_warnings”: [], “realm”: “”, “downtimes”: {}, “hostgroup_members”: [], “members”: [

“553d47bc-27aa-426c-a664-49c4c0c4a249”, “f88093ca-e61b-43ff-a41e-613f7ad2cea2”, “df1e2e13-552d-43de-ad2a-fe80ad4ba979”, “d3d667dd-f583-4668-9f44-22ef3dcb53ad”

]

}

}

Parameters

- **o_type** (*str*) – searched object type
- **o_name** (*str*) – searched object name (or uuid)

Returns serialized object information

Return type str

put_results()

Put results to scheduler, used by poller or reactionner when they are in active mode (passive = False)

This function is not intended for external use. Let the poller and reactionner manage all this stuff by themselves ;)

Parameters

- **from** (*str*) – poller/reactionner identification
- **results** (*list*) – list of actions results

Returns True

Return type bool

Module contents

This module provide all class for Alignak HTTP communication. Most of files are for server side HTTP. Only client.py is for client side http

2.1.4 alignak.misc package

Submodules

alignak.misc.carboniface module

External source code - note that all the methods in this class are not used by Alignak!

class alignak.misc.carboniface.**CarbonIface**(*host*, *port*, *event_url=None*)

Bases: object

add_data(*metric*, *value*, *ts=None*)

Add data to queue

Parameters

- **metric** (*str*) – the metric name
- **value** (*int*) – the value of data
- **ts** (*int | None*) – the timestamp

Returns True if added successfully, otherwise False

Return type bool

add_data_dict(*dd*)

dd must be a dictionary where keys are the metric name, each key contains a dictionary which at least must have 'value' key (optionally 'ts')

dd = {'experiment1.subsystem.block.metric1': {'value': 12.3, 'ts': 1379491605.55},

'experiment1.subsystem.block.metric2': {'value': 1.35},

... }

add_data_list(*dl*)

dl must be a list of tuples like: dl = [('metricname', (timestamp, value)),

('metricname', (timestamp, value)), ...]

add_event(*what*, *data=None*, *tags=None*, *when=None*)

Parameters

- **what** –
- **data** –
- **tags** –
- **when** –

Returns

send_data(*data=None*)

If data is empty, current buffer is sent. Otherwise data must be like: data = [('metricname', (timestamp, value)),

('metricname', (timestamp, value)), ...]

alignak.misc.common module

This module is used for common variables in Alignak. Previously some of those variables were linked to a specific class which made no sense.

class alignak.misc.common.**ModAttr**(*modattr*, *attribute*, *value*)

Bases: tuple

attribute

Alias for field number 1

modattr

Alias for field number 0

value

Alias for field number 2

alignak.misc.custom_module module

This module provides CustomModule class. Used to customize a module namespace

class alignak.misc.custom_module.**CustomModule**(*name*, *orig_mod_globals*)

Bases: module

Custom module that can be used to customize a module namespace,

example usage:

```
>>> import sys
>>> assert __name__ == 'custom_module'  # required for the import after
>>> class MyCustomModule(CustomModule):
...     count = 0
...     @property
...     def an_attribute(self):
...         self.count += 1
...         return "hey ! I'm a module attribute but also a property !"
>>> sys.modules[__name__] = MyCustomModule(__name__, globals())
```

then, in another module: >>> import custom_module >>> assert custom_module.count == 0 >>> custom_module.an_attribute "hey ! I'm a module attribute but also a property !" >>> assert custom_module.count == 1

alignak.misc.perfdata module

This module provide classes to handle performance data from monitoring plugin output

class alignak.misc.perfdata.**Metric**(*string*)
Bases: object

Class providing a small abstraction for one metric of a Perfdatas class

class alignak.misc.perfdata.**PerfDatas**(*string*)
Bases: object

Class providing performance data extracted from a check output

alignak.misc.perfdata.**guess_int_or_float**(*val*)
Wrapper for Util.to_best_int_float Basically cast into float or int and compare value If they are equal then there is no coma so return integer

Parameters **val** – value to cast

Returns value casted into int, float or None

Return type int | float | NoneType

alignak.misc.perfdata.**sanitize_name**(*field_name*)
Sanitize a field name for a TSDB (Graphite or Influx) - remove not allowed characters from the field name and replace with authorized characters

Parameters **field_name** (*string*) – Field name to clean

Returns sanitized field name

alignak.misc.serialization module

This module provide object serialization for Alignak objects. It basically converts objects to json

exception alignak.misc.serialization.**AlignakClassLookupException**
Bases: exceptions.Exception

Class for exceptions occurring in get_alignak_class from alignak.misc.serialization

alignak.misc.serialization.**get_alignak_class**(*python_path*)
Get the alignak class the in safest way I could imagine. Return None if (cumulative conditions)

```
* the module does not start with alignak
* above is false and the module is not is sys.modules
* above is false and the module does not have the wanted class
* above is false and the class in not a ClassType
```

Parameters **python_path** (*str*) –

Returns alignak class

:raise AlignakClassLookupException

alignak.misc.serialization.**serialize**(*obj*, *no_dump=False*)

Serialize an object.

Returns a dict containing an *_error* property if a MemoryError happens during the object serialization. See #369.

Parameters

- **obj** (*alignak.objects.item.Item | dict | list | str*) – the object to serialize
- **no_dump** (*bool*) – if True return dict, otherwise return a json

Returns

dict or json dumps dict with the following structure

```
{'__sys_python_module__': "%s.%s" % (o_cls.__module__, o_cls.__name__
↪)
```

'content' : obj.serialize()}

Return type dict | str

alignak.misc.serialization.**unserialize**(*j_obj*, *no_load=False*)

Un-serialize object. If we have __sys_python_module__ we try to safely get the alignak class Then we re-instantiate the alignak object

Parameters

- **j_obj** (*str (before loads)*) – json object, dict
- **no_load** (*bool*) – if True, j_obj is a dict, otherwise it's a json and need loads it

Returns un-serialized object

Module contents

Init of alignak.misc, nothing to do here.

2.1.5 alignak.modules package

Submodules

alignak.modules.inner_retention module

This module is used to manage retention in json files

class alignak.modules.inner_retention.**InnerRetention**(*mod_conf*)
Bases: *alignak.basemodule.BaseModule*

This class is used to store/restore retention data

do_loop_turn()
This function is called/used when you need a module with a loop function (and use the parameter 'external': True)

hook_load_retention(*scheduler*)
Load retention data from a file

Parameters **scheduler** (*object*) – scheduler instance of alignak

Returns None

hook_save_retention(*scheduler*)
Save retention data to a Json formated file

Parameters **scheduler** (*object*) – scheduler instance of alignak

Returns None

alignak.modules.inner_retention.**get_instance**(*mod_conf*)
Return a module instance for the modules manager

Parameters **mod_conf** – the module properties as defined globally in this file

Returns

Module contents

The modules package contains the definition of the Alignak inner defined modules.

Alignak inner defined modules are built for two reasons: - keep the Alignak modules logic (attach a module to a daemon to enhance its features) - provide some simple and almost mandatory features but not implement them in the application core

2.1.6 alignak.objects package

Submodules

alignak.objects.arbiterlink module

This module provide ArbiterLink and ArbiterLinks classes used to manage link with Arbiter daemon

class alignak.objects.arbiterlink.**ArbiterLink**(*params=None*, *parsing=True*)
Bases: *alignak.objects.satellitelink.SatelliteLink*

Class to manage the link to Arbiter daemon. With it, a master arbiter can communicate with a spare Arbiter daemon

do_not_run()
Check if satellite running or not If not, try to run

Returns true if satellite not running

Return type bool

is_me()
Check if parameter name if same than name of this object

TODO: is it useful?

Returns true if parameter name if same than this name

Return type bool

my_type = 'arbiter'

properties = {'active': <Property <class 'alignak.property.BoolProp'>, default: True

class alignak.objects.arbiterlink.**ArbiterLinks**(*items*, *index_items=True*, *parsing=True*)
Bases: *alignak.objects.satellitelink.SatelliteLinks*

Class to manage list of ArbiterLink. ArbiterLinks is used to regroup all links with Arbiter daemon

inner_class
alias of *ArbiterLink*

linkify(*modules=None*)
Link modules to Arbiter

TODO: why having this specific method? Because of this, Arbiters do not link with realms!

Parameters

- **realms** (*list*) – Realm object list (always None for an arbiter)
- **modules** (*list*) – list of modules

Returns None

name_property = 'arbiter_name'

alignak.objects.brokerlink module

This module provide BrokerLink and BrokerLinks classes used to manage brokers

class alignak.objects.brokerlink.**BrokerLink**(*params=None*, *parsing=True*)
Bases: *alignak.objects.satellitelink.SatelliteLink*

Class to manage the broker information

my_type = 'broker'

prepare_for_conf()
Initialize the pushed configuration dictionary with the inner properties that are to be propagated to the satellite link.

Returns None

properties = {'active': <Property <class 'alignak.property.BoolProp'>, default: True

class alignak.objects.brokerlink.**BrokerLinks**(*items*, *index_items=True*, *parsing=True*)
Bases: *alignak.objects.satellitelink.SatelliteLinks*

Class to manage list of BrokerLink. BrokerLinks is used to regroup all brokers

inner_class
alias of *BrokerLink*

name_property = 'broker_name'

alignak.objects.businessimpactmodulation module

This module provide Businessimpactmodulation and Businessimpactmodulations classes used to describe the modulation of a business impact. Modulation occurs on a modulation period (Timeperiod)

class alignak.objects.businessimpactmodulation.**Businessimpactmodulation**(*params=None*, *parsing=True*)

Bases: *alignak.objects.item.Item*

Businessimpactmodulation class is simply a modulation of the business impact value (of a Host/Service) during a modulation period.

get_name()

Accessor to business_impact_modulation_name attribute

Returns business impact modulation name

Return type str

my_type = 'businessimpactmodulation'

properties = {'business_impact': <Property <class 'alignak.property.IntegerProp'>, de

class alignak.objects.businessimpactmodulation.**Businessimpactmodulations**(*items*, *index_items=True*, *parsing=True*)

Bases: *alignak.objects.item.Items*

Businessimpactmodulations class allowed to handle easily several Businessimpactmodulation objects

inner_class

alias of *Businessimpactmodulation*

linkify(*timeperiods*)

Wrapper for Businessimpactmodulations.linkify_cm_by_tp(timeperiods)

Parameters **timeperiods** (alignak.objects.timeperiod.Timeperiods) – timeperiods to link to

Returns None

name_property = 'business_impact_modulation_name'

alignak.objects.checkmodulation module

This module provide CheckModulation and CheckModulations classes used to describe the modulation of a check command. Modulation occurs on a check period (Timeperiod)

class alignak.objects.checkmodulation.**CheckModulation**(*params=None*, *parsing=True*)

Bases: *alignak.objects.item.Item*

CheckModulation class is simply a modulation of the check command (of a Host/Service) during a check_period.

get_check_command(*timeperiods*, *t_to_go*)

Get the check_command if we are in the check period modulation

Parameters **t_to_go** – time to check if we are in the timeperiod

Returns A check command if we are in the check period, None otherwise

Return type *alignak.objects.command.Command*

get_name()
Accessor to checkmodulation_name attribute

Returns check modulation name

Return type str

is_correct()
Check if this object configuration is correct

```
* Check our own specific properties
* Call our parent class is_correct checker
```

Returns True if the configuration is correct, otherwise False

Return type bool

macros = {}

my_type = 'checkmodulation'

properties = {'check_command': <Property <class 'alignak.property.StringProp'>, defau

running_properties = {'conf_is_correct': <Property <class 'alignak.property.BoolProp'

serialize()
This function serialize into a simple dict object. It is used when transferring data to other daemons over the network (http)

Here is the generic function that simply export attributes declared in the properties dictionary and the running_properties of the object.

Returns Dictionary containing key and value from properties and running_properties

Return type dict

special_properties = ('check_period',)

class alignak.objects.checkmodulation.**CheckModulations**(*items*, *index_items=True*, *parsing=True*)
Bases: *alignak.objects.commandcallitem.CommandCallItems*

CheckModulations class allowed to handle easily several CheckModulation objects

inner_class
alias of *CheckModulation*

linkify(*timeperiods*, *commands*)
Replace check_period by real Timeperiod object into each CheckModulation Replace check_command by real Command object into each CheckModulation

Parameters

- **timeperiods** (alignak.objects.timeperiod.Timeperiods) – timeperiods to link to
- **commands** (alignak.objects.command.Commands) – commands to link to

Returns None

name_property = 'checkmodulation_name'

new_inner_member(*name=None*, *params=None*)
Create a CheckModulation object and add it to items

Parameters

- **name** (*str*) – CheckModulation name
- **params** (*dict*) – parameters to init CheckModulation

Returns None

TODO: Remove this default mutable argument. Usually result in unexpected behavior

alignak.objects.command module

This module provide Command class used to define external commands to check if something is ok or not

class alignak.objects.command.**Command**(*params=None*, *parsing=True*)
Bases: *alignak.objects.item.Item*

Class to manage a command A command is an external command that a poller module runs to check if something is ok or not

command_line

command_name

definition_order

enable_environment_macros

fill_data_brok_from(*data*, *brok_type*)
Add properties to data if fill_brok of these class properties is same as brok_type

Parameters

- **data** (*dict*) – dictionnary of this command
- **brok_type** (*str*) – type of brok

Returns None

get_name()
Get the name of the command

Returns the command name string

Return type str

imported_from

is_correct()
Check if this object configuration is correct

```
* Check our own specific properties
* Call our parent class is_correct checker
```

Returns True if the configuration is correct, otherwise False

Return type bool

module_type

my_type = 'command'

name

poller_tag

properties = {'command_line': <Property <class 'alignak.property.StringProp'>, defaul

reactionner_tag

register

timeout

use

class alignak.objects.command.**Commands**(*items*, *index_items=True*, *parsing=True*)
Bases: *alignak.objects.item.Items*

Class to manage all commands A command is an external command the poller module run to see if something is ok or not

inner_class
alias of *Command*

name_property = 'command_name'

alignak.objects.commandcallitem module

This module contains only a class for items objects that contains CommandCall objects.

class alignak.objects.commandcallitem.**CommandCallItems**(*items*, *index_items=True*, *parsing=True*)
Bases: *alignak.objects.item.Items*

This class provide simple methods to linkify CommandCall object. Only object that have CommandCall attribute need those methods (so no need to define it in Item)

static create_commandcall(*prop*, *commands*, *command*)
Create CommandCall object with command

Parameters

- **prop** (*str*) – property
- **commands** (alignak.objects.command.Commands) – all commands
- **command** (*str*) – a command object

Returns a commandCall object

Return type alignak.objects.commandcallitem.CommandCall

linkify_command_list_with_commands(*commands*, *prop*)
Link a command list (commands with , between) in real CommandCalls

Parameters

- **commands** (alignak.objects.command.Commands) – commands object
- **prop** (*str*) – property name

Returns None

linkify_one_command_with_commands(*commands*, *prop*)
Link a command to a property (check_command for example)

Parameters

- **commands** (alignak.objects.command.Commands) – commands object
- **prop** (*str*) – property name
- **default** (*str*) – default command to use if the property is not defined

Returns None

alignak.objects.config module

Config is the class that reads, loads and manipulates the main Alignak monitored objects configuration. It reads the Nagios legacy configuration files (cfg files) and gets all informations from these files.

It creates the monitored objects (eg. hosts, contacts, ...), creates links between them, check them, clean them, and cut them into independent parts.

The main user of this Config class is the Arbiter daemon when it loads the configuration and dispatches to the other daemons.

class alignak.objects.config.**Config**(*params=None*, *parsing=True*)

Bases: *alignak.objects.item.Item*

Config is the class that reads, loads and manipulates the main Alignak monitored objects configuration. It reads the Nagios legacy configuration files (cfg files) and gets all informations from these files.

It creates the monitored objects (eg. hosts, contacts, ...), creates links between them, check them, clean them, and cut them into independent parts.

The main user of this Config class is the Arbiter daemon when it loads the configuration and dispatches to the other daemons.

static add_self_defined_objects(*raw_objects*)

Add self defined command objects for internal processing ; bp_rule, _internal_host_up, _echo, _internal_host_check, _interna_service_check

Parameters **raw_objects** (*dict*) – Raw config objects dict

Returns raw_objects with some more commands

Return type dict

apply_dependencies()

Creates dependencies links between elements.

Returns None

apply_implicit_inheritance()

Wrapper for calling apply_implicit_inheritance method of services attributes Implicit inheritance is between host and service (like notification parameters etc)

:return:None

apply_inheritance()

Apply inheritance over templates Template can be used in the following objects:

```
* hosts
* contacts
* services
* servicedependencies
* hostdependencies
* timeperiods
* hostsextinfo
```

(continues on next page)

(continued from previous page)

```
* servicesextinfo
* serviceescalations
* hostescalations
* escalations
```

Returns None

cache_path = 'objects.cache'

check_error_on_hard_unmanaged_parameters()
Some parameters are just not managed like O*HP commands and regexp capabilities

Returns True if we encounter an error, otherwise False

Return type bool

clean()
Wrapper for calling the clean method of services attribute

Returns None

clean_params(*params*)
Convert a list of parameters (key=value) into a dict

This function is used to transform Nagios (or ini) like formated parameters (key=value) to a dictionary.

Parameters **params** (*list*) – parameters list

Returns dict with key and value. Log error if malformed

Return type dict

configuration_types = ['void', 'timeperiod', 'command', 'realm', 'host', 'hostgroup',

create_business_rules()
Create business rules for hosts and services

Returns None

create_business_rules_dependencies()
Create business rules dependencies for hosts and services

Returns None

create_objects(*raw_objects*)
Create all the objects got after the post configuration file initialization

Parameters **raw_objects** (*dict*) – dict with all object with str values

Returns None

create_objects_for_type(*raw_objects*, *o_type*)
Generic function to create objects regarding the o_type

This function create real Alignak objects from the raw data got from the configuration.

Parameters

- **raw_objects** (*dict*) – Raw objects
- **o_type** (*object*) – the object type we want to create

Returns None

create_packs()
Create packs of hosts and services (all dependencies are resolved) It create a graph. All hosts are connected to their parents, and hosts without parent are connected to host 'root'. services are linked to their host. Dependencies between hosts/services are managed. REF: doc/pack-creation.png

Returns None

cut_into_parts()
Cut conf into part for scheduler dispatch.

Basically it provides a set of host/services for each scheduler that have no dependencies between them

Returns None

dump(*dump_file_name=None*)
Dump configuration to a file in a JSON format

Parameters **dump_file_name** (*str*) – the file to dump configuration to

Returns None

early_arbiter_linking(*arbiter_name*, *params*)
Prepare the arbiter for early operations

Parameters **arbiter_name** (*str*) – default arbiter name if no arbiter exist in the configuration

Returns None

early_create_objects(*raw_objects*)
Create the objects needed for the post configuration file initialization

Parameters **raw_objects** (*dict*) – dict with all object with str values

Returns None

early_created_types = ['arbiter', 'module']

explode()
Use to fill groups values on hosts and create new services (for host group ones)

Returns None

explode_global_conf()
Explode parameters like cached_service_check_horizon in the Service class in a cached_check_horizon manner, o*hp commands etc

Returns None

fill_default_configuration()
Fill objects properties with default value if necessary

Returns None

fill_default_realm()
Check if a realm is defined, if not Create a new one (default) and tag everyone that do not have a realm prop to be put in this realm

Returns None

fill_default_satellites(*alignak_launched=False*)
If a required satellite is missing in the configuration, we create a new satellite on localhost with some default values

Parameters **alignak_launched** (*bool*) – created daemons are to be launched or not

Returns None

got_arbiter_module_type_defined(*module_type*)
Check if a module type is defined in one of the arbiters Also check the module name

Parameters **module_type** (*str*) – module type to search for

Returns True if mod_type is found else False

Return type bool

TODO: Factorize it with got_broker_module_type_defined:

got_broker_module_type_defined(*module_type*)
Check if a module type is defined in one of the brokers

Parameters **module_type** (*str*) – module type to search for

Returns True if mod_type is found else False

Return type bool

got_scheduler_module_type_defined(*module_type*)
Check if a module type is defined in one of the schedulers

Parameters **module_type** (*str*) – module type to search for

Returns True if mod_type is found else False

Return type bool

TODO: Factorize it with got_broker_module_type_defined

hack_old_nagios_parameters()
Check if modules exist for some of the Nagios legacy parameters.

If no module of the required type is present, it alerts the user that the parameters will be ignored and the functions will be disabled, else it encourages the user to set the correct parameters in the installed modules.

Note that some errors are raised if some parameters are used and no module is found to manage the corresponding feature.

TODO: clean this part of the configuration checking! Nagios ascending compatibility!

Returns modules list

Return type list

is_correct()
Check if all elements got a good configuration

Returns True if the configuration is correct else False

Return type bool

linkify()
Make 'links' between elements, like a host got a services list with all its services in it

Returns None

linkify_one_command_with_commands(*commands*, *prop*)
Link a command

Parameters

- **commands** (*object*) – object commands

- **prop** (*str*) – property name

Returns None

linkify_templates()
Like for normal object, we link templates with each others

Returns None

load_params(*params*)
Load parameters from main configuration file

Parameters **params** – parameters list (converted right at the beginning)

Returns None

log_daemons_list()
Log Alignak daemons list

Returns

macros = {'ADMINEMAIL': '', 'ADMINPAGER': '', 'ALIGNAK': 'alignak_name', 'ALIGNAK_CONF

my_type = 'config'
Configuration properties:

old_properties = {'nagios_group': 'alignak_group', 'nagios_user': 'alignak_user'}

override_properties()
Wrapper for calling override_properties method of services attribute

Returns

prepare_for_sending()
The configuration needs to be serialized before being sent to a spare arbiter

Returns None

propagate_timezone_option()
Set our timezone value and give it too to unset satellites

Returns None

> />, 'passive_hos

read_config_buf(*cfg_buffer*)
The legacy configuration buffer (previously returned by Config.read_config())

If the buffer is empty, it will return an empty dictionary else it will return a dictionary containing dictionary items tha tmay be used to create Alignak objects

Parameters **cfg_buffer** (*str*) – buffer containing all data from config files

Returns dict of alignak objects with the following structure

{ type1 [[{key: value, ..}, {..}],] type2 : [...]

}

Example

```
{ 'host' : [{'host_name': 'myhostname', ..}, {..}],
```

'service' : [...]

}

Values are all str for now. It is pythonized at object creation

Return type dict

read_config_silent = False

read_legacy_cfg_files(*cfg_files*, *alignak_env_files=None*)

Read and parse the Nagios legacy configuration files and store their content into a StringIO object which content will be returned as the function result

Parameters

- **cfg_files** (*list*) – list of file to read
- **alignak_env_files** (*list*) – name of the alignak environment file

Returns a buffer containing all files

Return type str

remove_templates()

Clean useless elements like templates because they are not needed anymore

Returns None

serialize()

This function serialize into a simple dict object. It is used when transferring data to other daemons over the network (http)

Here is the generic function that simply export attributes declared in the properties dictionary and the running_properties of the object.

Returns Dictionary containing key and value from properties and running_properties

Return type dict

show_errors()

Loop over configuration warnings and log them as INFO log Loop over configuration errors and log them as INFO log

Note that the warnings and errors are logged on the fly during the configuration parsing. It is not necessary to log as WARNING and ERROR in this function which is used as a sum-up on the end of configuration parsing when an error has been detected.

Returns None

types_creations = {'arbiter': (<class 'alignak.objects.arbiterlink.ArbiterLink'>, <cl

warn_about_unmanaged_parameters()

used to raise warning if the user got parameter that we do not manage from now

Returns None

alignak.objects.contact module

This module provide Contact and Contacts classes that implements contact for notification. Basically used for parsing.

class alignak.objects.contact.**Contact**(*params=None*, *parsing=True*)

Bases: *alignak.objects.item.Item*

Host class implements monitoring concepts for contact. For example it defines host_notification_period, service_notification_period etc.

get_groupname()
Get the first group name whose contact belongs to :return: group name :rtype: str

get_groupnames()
Get all the groups name whose contact belongs to :return: comma separated list of the groups names :rtype: str

get_name()
Get contact name

Returns contact name

Return type str

get_notification_commands(*notifways*, *n_type*, *command_name=False*)
Get notification commands for object type

Parameters

- **notifways** (NotificationWays) – list of alignak.objects.NotificationWay objects
- **n_type** (*string*) – object type (host or service)
- **command_name** (*bool*) – True to update the inner property with the name of the command, False to update with the Command objects list

Returns command list

Return type list[*alignak.objects.command.Command*]

is_correct()
Check if this object configuration is correct

```
* Check our own specific properties
* Call our parent class is_correct checker
```

Returns True if the configuration is correct, otherwise False

Return type bool

macros = {'CONTACTADDRESS1': 'address1', 'CONTACTADDRESS2': 'address2', 'CONTACTADDR

my_type = 'contact'

old_properties = {'min_criticity': 'min_business_impact'}

properties = {'address1': <Property <class 'alignak.property.StringProp'>, default:

raise_cancel_downtime_log_entry()
Raise CONTACT DOWNTIME ALERT entry (info level) Format is : "CONTACT DOWNTIME ALERT: *get_name()*;CANCELLED;

Contact has entered a period of scheduled downtime"

Example ["CONTACT DOWNTIME ALERT: test_contact;CANCELLED;] Contact has entered a period of scheduled downtime"

Returns None

raise_enter_downtime_log_entry()
Raise CONTACT DOWNTIME ALERT entry (info level) Format is : "CONTACT DOWNTIME ALERT: *get_name()*;STARTED;

Contact has entered a period of scheduled downtime”

Example [“CONTACT DOWNTIME ALERT: test_contact;STARTED;] Contact has entered a period of scheduled downtime”

Returns None

raise_exit_downtime_log_entry()
Raise CONTACT DOWNTIME ALERT entry (info level) Format is : “CONTACT DOWNTIME ALERT: *get_name()*;STOPPED;

Contact has entered a period of scheduled downtime”

Example [“CONTACT DOWNTIME ALERT: test_contact;STOPPED;] Contact has entered a period of scheduled downtime”

Returns None

running_properties = {'broks': <Property <class 'alignak.property.ListProp'>, default

serialize()
This function serialize into a simple dict object. It is used when transferring data to other daemons over the network (http)

Here is the generic function that simply export attributes declared in the properties dictionary and the running_properties of the object.

Returns Dictionary containing key and value from properties and running_properties

Return type dict

simple_way_parameters = ('service_notification_period', 'host_notification_period', 's

special_properties = ('service_notification_commands', 'host_notification_commands', '

want_host_notification(*notifways*, *timeperiods*, *timestamp*, *state*, *n_type*, *business_impact*, *cmd=None*)
Check if notification options match the state of the host

Parameters

- **timestamp** (*int*) – time we want to notify the contact (usually now)
- **state** (*str*) – host or service state (“UP”, “DOWN” ..)
- **n_type** (*str*) – type of notification (“PROBLEM”, “RECOVERY” ..)
- **business_impact** (*int*) – impact of this host
- **cmd** (*str*) – command launch to notify the contact

Returns True if contact wants notification, otherwise False

Return type bool

want_service_notification(*notifways*, *timeperiods*, *timestamp*, *state*, *n_type*, *business_impact*, *cmd=None*)
Check if notification options match the state of the service

Parameters

- **timestamp** (*int*) – time we want to notify the contact (usually now)
- **state** (*str*) – host or service state (“WARNING”, “CRITICAL” ..)

- **n_type** (*str*) – type of notification ("PROBLEM", "RECOVERY" ..)
- **business_impact** (*int*) – impact of this service
- **cmd** (*str*) – command launched to notify the contact

Returns True if contact wants notification, otherwise False

Return type bool

class alignak.objects.contact.**Contacts**(*items*, *index_items=True*, *parsing=True*)

Bases: *alignak.objects.commandcallitem.CommandCallItems*

Contacts manage a list of Contacts objects, used for parsing configuration

explode(*contactgroups*, *notificationways*)

Explode all contact for each contactsgroup

Parameters

- **contactgroups** (alignak.objects.contactgroup.Contactgroups) – contactgroups to explode
- **notificationways** (*alignak.objects.notificationway.Notificationways*) – notificationways to explode

Returns None

inner_class

alias of *Contact*

linkify(*commands*, *notificationways*)

Create link between objects:

```
* contacts -> notificationways
```

Parameters **notificationways** (*alignak.objects.notificationway.Notificationways*) – notificationways to link

Returns None

TODO: Clean this function

linkify_with_notificationways(*notificationways*)

Link hosts with realms

Parameters **notificationways** (*alignak.objects.notificationway.Notificationways*) – notificationways object to link with

Returns None

name_property = 'contact_name'

alignak.objects.contactgroup module

This module provide Contactgroup and Contactgroups class used to manage contact groups

class alignak.objects.contactgroup.**Contactgroup**(*params=None*, *parsing=True*)

Bases: *alignak.objects.itemgroup.Itemgroup*

Class to manage a group of contacts A Contactgroup is used to manage a group of contacts

get_contactgroup_members()

Get the groups members of the group

Returns list of contacts

Return type list

get_contacts()
Get the contacts of the group

Returns list of contacts

Return type list[*alignak.objects.contact.Contact*]

get_contacts_by_explosion(*contactgroups*)
Get contacts of this group

Parameters **contactgroups** (alignak.objects.contactgroup.Contactgroups) – Contactgroups object, use to look for a specific one

Returns list of contact of this group

Return type list[*alignak.objects.contact.Contact*]

get_name()
Get the group name

group_members_property = 'contactgroup_members'

macros = {'CONTACTGROUPALIAS': 'alias', 'CONTACTGROUPGROUPMEMBERS': 'get_contactgroup_

members_property = 'members'

my_type = 'contactgroup'

properties = {'alias': <Property <class 'alignak.property.StringProp'>, default: u''

class alignak.objects.contactgroup.**Contactgroups**(*items*, *index_items=True*, *parsing=True*)
Bases: *alignak.objects.itemgroup.Itemgroups*

Class to manage list of Contactgroup Contactgroups is used to regroup all Contactgroup

add_contactgroup(*contactgroup*)
Wrapper for add_item method Add a contactgroup to the contactgroup list

Parameters **contactgroup** – contact group to add

Returns None

add_member(*contact_name*, *contactgroup_name*)
Add a contact string to a contact member if the contact group do not exist, create it

Parameters

- **contact_name** (*str*) – contact name
- **contactgroup_name** (*str*) – contact group name

Returns None

explode()
Fill members with contactgroup_members

:return:None

get_members_of_group(*gname*)
Get all members of a group which name is given in parameter

Parameters **gname** (*str*) – name of the group

Returns list of contacts in the group

Return type list[*alignak.objects.contact.Contact*]

inner_class
alias of *Contactgroup*

linkify(*contacts*)
Create link between objects:

```
* contactgroups -> contacts
```

Parameters **contacts** (alignak.objects.contact.Contacts) – contacts to link

Returns None

linkify_contactgroups_contacts(*contacts*)
Link the contacts with contactgroups

Parameters **contacts** (alignak.objects.contact.Contacts) – realms object to link with

Returns None

name_property = 'contactgroup_name'

alignak.objects.escalation module

This module provides Escalation and Escalations classes that implements escalation for notification. Basically used for parsing.

class alignak.objects.escalation.**Escalation**(*params=None*, *parsing=True*)
Bases: *alignak.objects.item.Item*

Escalation class is used to implement notification escalation

get_name()
Accessor to escalation_name attribute

Returns escalation name

Return type str

get_next_notif_time(*t_wished*, *status*, *creation_time*, *interval*, *escal_period*)
Get the next notification time for the escalation Only legit for time based escalation

Parameters

- **t_wished** – time we would like to send a new notification (usually now)
- **status** – status of the host or service
- **creation_time** – time the notification was created
- **interval** (*int*) – time interval length

Returns timestamp for next notification or None

Return type int | None

is_correct()
Check if this object configuration is correct

```
* Check our own specific properties
* Call our parent class is_correct checker
```

Returns True if the configuration is correct, otherwise False

Return type bool

is_eligible(*timestamp*, *status*, *notif_number*, *in_notif_time*, *interval*, *escal_period*)
Check if the escalation is eligible (notification is escalated or not)

Escalation is NOT eligible in ONE of the following condition is fulfilled:

```
* escalation is not time based and notification number not in range
```

[first_notification;last_notification] (if last_notif == 0, it's infinity)

- escalation is time based and notification time not in range [first_notification_time;last_notification_time] (if last_notif_time == 0, it's infinity)
- status does not matches escalation_options ('WARNING' <=> 'w' ...)
- escalation_period is not legit for this time (now usually)

Parameters

- **timestamp** (*int*) – timestamp to check if timeperiod is valid
- **status** (*str*) – item status (one of the small_states key)
- **notif_number** (*int*) – current notification number
- **in_notif_time** (*int*) – current notification time
- **interval** (*int*) – time interval length

Returns True if no condition has been fulfilled, otherwise False

Return type bool

my_type = 'escalation'

properties = {'contact_groups': <Property <class 'alignak.property.ListProp'>, defaul

running_properties = {'conf_is_correct': <Property <class 'alignak.property.BoolProp'

special_properties = ('contacts', 'contact_groups', 'first_notification_time', 'last_n

special_properties_time_based = ('contacts', 'contact_groups', 'first_notification', '

class alignak.objects.escalation.**Escalations**(*items*, *index_items=True*, *parsing=True*)
Bases: *alignak.objects.item.Items*

Escalations manage a list of Escalation objects, used for parsing configuration

add_escalation(*escalation*)
Wrapper for add_item method

Parameters **escalation** (alignak.objects.escalation.Escalation) – escalation to add to item dict

Returns None

explode(*hosts*, *hostgroups*, *contactgroups*)
Loop over all escalation and explode hostsgroups in host and contactgroups in contacts

Call Item.explode_host_groups_into_hosts and Item.explode_contact_groups_into_contacts

Parameters

- **hosts** (alignak.objects.host.Hosts) – host list to explode
- **hostgroups** (alignak.objects.hostgroup.Hostgroups) – hostgroup list to explode
- **contactgroups** (alignak.objects.contactgroup.Contactgroups) – contactgroup list to explode

Returns None

inner_class
alias of *Escalation*

linkify (*timeperiods*, *contacts*, *services*, *hosts*)
Create link between objects:

```
* escalation -> host
* escalation -> service
* escalation -> timeperiods
* escalation -> contact
```

Parameters

- **timeperiods** (alignak.objects.timeperiod.Timeperiods) – timeperiods to link
- **contacts** (alignak.objects.contact.Contacts) – contacts to link
- **services** (alignak.objects.service.Services) – services to link
- **hosts** (alignak.objects.host.Hosts) – hosts to link

Returns None

linkify_es_by_h (*hosts*)
Add each escalation object into host.escalation attribute

Parameters **hosts** (alignak.objects.host.Hosts) – host list, used to look for a specific host

Returns None

linkify_es_by_s (*services*)
Add each escalation object into service.escalation attribute

Parameters **services** (alignak.objects.service.Services) – service list, used to look for a specific service

Returns None

name_property = 'escalation_name'

alignak.objects.genericextinfo module

This is the main class for the generic ext info. In fact it's mainly about the configuration part. Parameters are merged in Host or Service so it's no use in running part

class alignak.objects.genericextinfo.**GenericExtInfo** (*params=None*, *parsing=True*)
Bases: *alignak.objects.item.Item*

GenericExtInfo class is made to handle some parameters of SchedulingItem:

```
* notes
* notes_url
* icon_image
* icon_image_alt
```

get_name()
Accessor to host_name attribute or name if first not defined

Returns host name, use to search the host to merge

Return type str

alignak.objects.host module

This is the main class for the Host. In fact it's mainly about the configuration part. for the running one, it's better to look at the schedulingitem class that manage all scheduling/consume check smart things :)

class alignak.objects.host.**Host**(*params=None*, *parsing=True*)
Bases: *alignak.objects.schedulingitem.SchedulingItem*

Host class implements monitoring concepts for host. For example it defines parents, check_interval, check_command etc.

acknowledgement

acknowledgement_type

act_depend_of

act_depend_of_me

action_url

actions

active_checks_enabled

add_service_link(*service*)
Add a service to the service list of this host

Parameters **service** (alignak.objects.service.Service) – the service to add

Returns None

address

address6

alias

attempt

broks

business_impact

business_impact_modulations

business_rule

business_rule_downtime_as_ack

business_rule_host_notification_options

business_rule_output_template

business_rule_service_notification_options

business_rule_smart_notifications

check_command

check_flapping_recovery_notification

check_freshness

check_interval

check_period

checkmodulations

checks_in_progress

child_dependencies

chk_depend_of

chk_depend_of_me

comments

conf_is_correct

configuration_errors

configuration_warnings

contact_groups

contacts

static convert_conf_for_unreachable(*params*)

The 'u' state for UNREACHABLE has been rewritten in 'x' in: * flap_detection_options * notification_options * snapshot_criteria

So convert value from config file to keep compatibility with Nagios

Parameters **params** (*dict*) – parameters of the host before put in properties

Returns None

current_event_id

int(x=0) -> int or long int(x, base=10) -> int or long

Convert a number or string to an integer, or return 0 if no arguments are given. If x is floating point, the conversion truncates towards zero. If x is outside the integer range, the function returns a long instead.

If x is not a number or if base is given, then x must be a string or Unicode object representing an integer literal in the given base. The literal can be preceded by '+' or '-' and be surrounded by whitespace. The base defaults to 10. Valid bases are 0 and 2-36. Base 0 means to interpret the base from the string as an integer literal. >>> int('0b100', base=0) 4

current_notification_id

current_notification_number

current_problem_id

int(x=0) -> int or long int(x, base=10) -> int or long

Convert a number or string to an integer, or return 0 if no arguments are given. If x is floating point, the conversion truncates towards zero. If x is outside the integer range, the function returns a long instead.

If x is not a number or if base is given, then x must be a string or Unicode object representing an integer literal in the given base. The literal can be preceded by '+' or '-' and be surrounded by whitespace. The base defaults to 10. Valid bases are 0 and 2-36. Base 0 means to interpret the base from the string as an integer literal. >>> int('0b100', base=0) 4

custom_views

customs

definition_order

display_name

downtimes

duration_sec

early_timeout

end_time

escalations

event_handler

event_handler_enabled

execution_time

fill_predictive_missing_parameters()
Fill address with host_name if not already set and define state with initial_state

Returns None

first_notification_delay

flap_detection_enabled

flap_detection_options

flapping_changes

flapping_comment_id

freshness_expired

freshness_log_raised

freshness_state

freshness_threshold

get_ack_author_name()
Get the author of the acknowledgement

Returns author

Return type str

get_ack_comment()
Get the comment of the acknowledgement

Returns comment

Return type str

get_check_command()
Wrapper to get the name of the check_command attribute

Returns check_command name

Return type str

get_downtime()
Accessor to scheduled_downtime_depth attribute

Returns scheduled downtime depth

Return type str

get_duration()
Get duration formatted Format is : "HHh MMm SSs" Example : "10h 20m 40s"

Returns Formatted duration

Return type str

get_duration_sec()
Get duration in seconds. (cast it before returning)

Returns duration in seconds

Return type int

TODO: Move to util or SchedulingItem class

get_full_name()
Accessor to host_name attribute

Returns host_name

Return type str

get_groupalias(*hostgroups*)
Get alias of the first host's hostgroup (alphabetic sort on group alias)

Returns host group alias

Return type str

TODO: Clean this. It returns the first hostgroup alias (alphabetic sort)

get_groupaliases(*hostgroups*)
Get aliases of the host's hostgroups

Returns comma separated aliases of hostgroups alphabetically sorted

Return type str

get_groupname(*hostgroups*)
Get name of the first host's hostgroup (alphabetic sort)

Returns host group name

Return type str

TODO: Clean this. It returns the first hostgroup (alphabetic sort)

get_groupnames(*hostgroups*)
Get names of the host's hostgroups

Returns comma separated names of hostgroups alphabetically sorted

Return type str

get_hostgroups()
Accessor to hostgroups attribute

Returns hostgroup list object of host

Return type list

get_name()
Get the host name. Try several attributes before returning UNNAMED*

Returns The name of the host

Return type str

get_overall_state(*services*)
Get the host overall state including the host self status and the status of its services

Compute the host overall state identifier, including: - the acknowledged state - the downtime state

The host overall state is (prioritized): - an host not monitored (5) - an host down (4) - an host unreachable (3) - an host downtimed (2) - an host acknowledged (1) - an host up (0)

If the host overall state is <= 2, then the host overall state is the maximum value of the host overall state and all the host services overall states.

The overall state of an host is: - 0 if the host is UP and all its services are OK - 1 if the host is DOWN or UNREACHABLE and acknowledged or

at least one of its services is acknowledged and no other services are WARNING or CRITICAL

- **2 if the host is DOWN or UNREACHABLE and in a scheduled downtime or** at least one of its services is in a scheduled downtime and no other services are WARNING or CRITICAL
- **3 if the host is UNREACHABLE or** at least one of its services is WARNING
- **4 if the host is DOWN or** at least one of its services is CRITICAL
- 5 if the host is not monitored

Parameters **services** (alignak.objects.service.Services) – a list of known services

Returns the host overall state

Return type int

get_services()
Get all services for this host

Returns list of services

Return type list

get_short_status(*hosts*, *services*)
Get the short status of this host

Returns "U", "D", "X" or "n/a" based on host state_id or business_rule state

Return type str

get_snapshot_command()
Wrapper to get the name of the snapshot_command attribute

Returns snapshot_command name

Return type str

get_status(*hosts*, *services*)
Get the status of this host

Returns "UP", "DOWN", "UNREACHABLE" or "n/a" based on host state_id or business_rule state

Return type str

get_total_services()
Get the number of services for this host

Returns service list length

Return type str

get_total_services_critical(*services*)
Get number of services critical

Parameters **services** –

Returns Number of services

Return type int

get_total_services_ok(*services*)
Get number of services ok

Parameters **services** –

Returns Number of services

Return type int

get_total_services_unknown(*services*)
Get number of services unknown

Parameters **services** –

Returns Number of services

Return type int

get_total_services_unreachable(*services*)
Get number of services unreachable

Parameters **services** –

Returns Number of services

Return type int

get_total_services_warning(*services*)
Get number of services warning

Parameters **services** –

Returns Number of services

Return type int

got_business_rule

got_default_realm

has_been_checked

high_flap_threshold

host_name

hostgroups

icon_image

icon_image_alt

icon_set

impacts

imported_from

in_checking

in_hard_unknown_reach_phase

in_maintenance

in_scheduled_downtime

in_scheduled_downtime_during_last_check

initial_state

is_blocking_notifications(*notification_period*, *hosts*, *services*, *n_type*, *t_wished*)

Check if a notification is blocked by the host. Conditions are ONE of the following:

```
* enable_notification is False (global)
* not in a notification_period
* notifications_enable is False (local)
* notification_options is 'n' or matches the state ('DOWN' <=> 'd' ...)
```

(include flapping and downtimes)

- state goes ok and type is 'ACKNOWLEDGEMENT' (no sense)
- scheduled_downtime_depth > 0 and flapping (host is in downtime)
- scheduled_downtime_depth > 1 and not downtime end (deep downtime)
- scheduled_downtime_depth > 0 and problem or recovery (host is in downtime)
- SOFT state of a problem (we raise notification ony on HARD state)
- ACK notification when already ACK (don't raise again ACK)
- not flapping notification in a flapping state
- business rule smart notifications is enabled and all its children have been acknowledged or are under downtime

Parameters

- **n_type** – notification type
- **t_wished** (*float*) – the time we should like to notify the host (mostly now)

Returns True if ONE of the above condition was met, otherwise False

Return type bool

TODO: Refactor this, a lot of code duplication with Service.is_blocking_notifications

is_correct()

Check if this object configuration is correct

```
* Check our own specific properties
* Call our parent class is_correct checker
```

Returns True if the configuration is correct, otherwise False

Return type bool

is_excluded_for(*service*)

Check whether this host should have the passed service be "excluded" or "not included".

An host can define service_includes and/or service_excludes directive to either white-list-only or black-list some services from itself.

Parameters **service** (alignak.objects.service.Service) –

Returns True if is excluded, otherwise False

Return type bool

is_excluded_for_sdesc(*sdesc*, *is_tpl=False*)

Check whether this host should have the passed service *description* be "excluded" or "not included".

Parameters

- **sdesc** – service description
- **is_tpl** (*bool*) – True if service is template, otherwise False

Returns True if service description excluded, otherwise False

Return type bool

is_flapping

is_impact

is_problem

is_state(*status*)

Return if status match the current host status

Parameters **status** (*str*) – status to compare ("o", "d", "x"). Usually comes from config files

Returns True if status <=> self.status, otherwise False

Return type bool

labels

last_check_command

last_chk

last_event_id

last_hard_state

last_hard_state_change

last_hard_state_id

last_notification

last_perf_data

last_problem_id

last_snapshot

last_state

last_state_change

last_state_id

last_state_type

last_state_update

last_time_down

last_time_non_ok_or_up()
Get the last time the host was in a non-OK state

Returns self.last_time_down if self.last_time_down > self.last_time_up, 0 otherwise

Return type int

last_time_unreachable

last_time_up

latency

long_output

low_flap_threshold

macromodulations

macros = {'FULLNAME': 'get_full_name', 'HOSTACKAUTHOR': 'get_ack_author_name', 'HOSTAC

maintenance_period

manage_stalking(*check*)
Check if the host need stalking or not (immediate recheck) If one stalking_options matches the exit_status ('o' <=> 0 . . .) then stalk is needed Raise a log entry (info level) if stalk is needed

Parameters **check** (alignak.check.Check) – finished check (check.status == 'wait-consume')

Returns None

max_check_attempts

modified_attributes

my_own_business_impact

my_type = 'host'

name

next_chk

notes

notes_url

notification_interval

notification_is_blocked_by_contact(*notifways*, *timeperiods*, *notif*, *contact*)
Check if the notification is blocked by this contact.

Parameters

- **notif** (alignak.objects.contact.Contact) – notification created earlier
- **contact** – contact we want to notify

Returns True if the notification is blocked, False otherwise

Return type bool

notification_options

notification_period

notifications_enabled

notifications_in_progress

notified_contacts

notified_contacts_ids

ok_up = u'UP'

old_properties = {'criticity': 'business_impact', 'hostgroup': 'hostgroups', 'normal

output

parent_dependencies

parents

passive_checks_enabled

pending_flex_downtime

percent_state_change

perf_data

poller_tag

problem_has_been_acknowledged

process_perf_data

processed_business_rule

properties = {'2d_coords': <Property <class 'alignak.property.StringProp'>, default:

raise_acknowledge_log_entry()
Raise HOST ACKNOWLEDGE ALERT entry (critical level)

Returns None

raise_alert_log_entry()
Raise HOST ALERT entry Format is : "HOST ALERT: *get_name()*;*state*;*state_type*;*attempt*;*output*"
Example : "HOST ALERT: server;DOWN;HARD;1;I don't know what to say..."

Returns None

raise_cancel_downtime_log_entry()
Raise HOST DOWNTIME ALERT entry (critical level) Format is : "HOST DOWNTIME ALERT: *get_name()*;CANCELLED;

Host has entered a period of scheduled downtime"

Example ["HOST DOWNTIME ALERT: test_host_0;CANCELLED;] Host has entered a period of scheduled downtime"

Returns None

raise_check_result()
Raise ACTIVE CHECK RESULT entry Example : "ACTIVE HOST CHECK: server;DOWN;HARD;1;I don't know what to say..."

Returns None

raise_enter_downtime_log_entry()
Raise HOST DOWNTIME ALERT entry (critical level) Format is : "HOST DOWNTIME ALERT: *get_name()*;STARTED;

Host has entered a period of scheduled downtime"

Example ["HOST DOWNTIME ALERT: test_host_0;STARTED;] Host has entered a period of scheduled downtime"

Returns None

raise_event_handler_log_entry(*command*)
Raise HOST EVENT HANDLER entry (critical level) Format is : "HOST EVENT HANDLER: *self.get_name()*;*state*;*state_type*;*attempt*;

command.get_name()"

Example : "HOST EVENT HANDLER: server;UP;HARD;1;notify-by-rss"

Parameters **command** (alignak.objects.command.Command) – Handler launched

Returns None

raise_exit_downtime_log_entry()
Raise HOST DOWNTIME ALERT entry (critical level) Format is : "HOST DOWNTIME ALERT: *get_name()*;STOPPED;

Host has entered a period of scheduled downtime"

Example ["HOST DOWNTIME ALERT: test_host_0;STOPPED;] Host has entered a period of scheduled downtime"

Returns None

raise_flapping_start_log_entry(*change_ratio*, *threshold*)
Raise HOST FLAPPING ALERT START entry (critical level) Format is : "HOST FLAPPING ALERT: *self.get_name()*;STARTED;

Host appears to have started flapping (*change_ratio*% change >= *threshold*% threshold)"

Example ["HOST FLAPPING ALERT: server;STARTED;] Host appears to have started flapping (50.6% change >= 50.0% threshold)"

Parameters

- **change_ratio** (*float*) – percent of changing state
- **threshold** (*float*) – threshold (percent) to trigger this log entry

Returns None

raise_flapping_stop_log_entry(*change_ratio*, *threshold*)
Raise HOST FLAPPING ALERT STOPPED entry (critical level) Format is : "HOST FLAPPING ALERT: *self.get_name()*;STOPPED;

Host appears to have stopped flapping (*change_ratio*% change < *threshold*% threshold)"

Example ["HOST FLAPPING ALERT: server;STOPPED;] Host appears to have stopped flapping (23.0% change < 25.0% threshold)"

Parameters

- **change_ratio** (*float*) – percent of changing state
- **threshold** (*float*) – threshold (percent) to trigger this log entry

Returns None

raise_initial_state()
Raise CURRENT HOST ALERT entry (info level) Format is : "CURRENT HOST STATE: *get_name()*;*state*;*state_type*;*attempt*;*output*" Example : "CURRENT HOST STATE: server;DOWN;HARD;1;I don't know what to say..."

Returns None

raise_no_next_check_log_entry()
Raise no scheduled check entry (warning level) Format is : "I cannot schedule the check for the host 'get_name()*'

because there is not future valid time"

Example ["I cannot schedule the check for the host 'Server'] because there is not future valid time"

Returns None

raise_notification_log_entry(*notif*, *contact*, *host_ref=None*)
Raise HOST NOTIFICATION entry (critical level) Format is : "HOST NOTIFICATION: *contact.get_name()*;*self.get_name()*;*state*;

command.get_name();*output*"

Example : "HOST NOTIFICATION: superadmin;server;UP;notify-by-rss;no output"

Parameters **notif** (*alignak.objects.notification.Notification*) – notification object created by host alert

Returns None

raise_snapshot_log_entry(*command*)
Raise HOST SNAPSHOT entry (critical level) Format is : "HOST SNAPSHOT: *self.get_name()*;*state*;*state_type*;*attempt*;

command.get_name()"

Example : "HOST SNAPSHOT: server;UP;HARD;1;notify-by-rss"

Parameters **command** (alignak.objects.command.Command) – Snapshot command launched

Returns None

raise_unacknowledge_log_entry()
Raise HOST ACKNOWLEDGE STOPPED entry (critical level)

Returns None

reactionner_tag

realm

realm_name

register

resultmodulations

retain_nonstatus_information

retain_status_information

retry_interval

return_code

running_properties = {'acknowledgement': <Property <class 'alignak.property.StringPro

s_time

scheduled_downtime_depth

service_excludes = []

service_includes = []

service_overrides

services

set_state_from_exit_status (*status*, *notif_period*, *hosts*, *services*)
Set the state in UP, DOWN, or UNREACHABLE according to the status of a check result.

Parameters **status** (*int*) – integer between 0 and 3 (but not 1)

Returns None

should_be_scheduled

snapshot_command

snapshot_criteria

snapshot_enabled

snapshot_interval

snapshot_period

source_problems

stalking_options

start_time

state

state_before_hard_unknown_reach_phase

state_before_impact

state_changed_since_impact

state_id

state_id_before_impact

state_type

state_type_id

statusmap_image

tags

time_to_orphanage

timeout

topology_change

trending_policies

u_time

use

vrml_image

was_in_hard_unknown_reach_phase

class alignak.objects.host.**Hosts**(*items*, *index_items=True*, *parsing=True*)
Bases: *alignak.objects.schedulingitem.SchedulingItems*

Class for the hosts lists. It's mainly for configuration

apply_dependencies()
Loop on hosts and register dependency between parent and son

call Host.fill_parents_dependency()

Returns None

explode(*hostgroups*, *contactgroups*)
Explode hosts with hostgroups, contactgroups:

```
* Add contact from contactgroups to host contacts
* Add host into their hostgroups as hostgroup members
```

Parameters

- **hostgroups** (alignak.objects.hostgroup.Hostgroups) – Hostgroups to explode
- **contactgroups** (alignak.objects.contactgroup.Contactgroups) – Contactgorups to explode

Returns None

fill_predictive_missing_parameters()
Loop on hosts and call Host.fill_predictive_missing_parameters()

Returns None

find_hosts_that_use_template(*tpl_name*)
Find hosts that use the template defined in argument tpl_name

Parameters **tpl_name** (*str*) – the template name we filter or

Returns list of the host_name of the hosts that got the template tpl_name in tags

Return type list[str]

inner_class
alias of *Host*

is_correct()
Check if the hosts list configuration is correct

```
* check if any loop exists in each host dependencies
* Call our parent class is_correct checker
```

Returns True if the configuration is correct, otherwise False

Return type bool

linkify(*timeperiods=None*, *commands=None*, *contacts=None*, *realms=None*, *resultmodulations=None*, *businessimpactmodulations=None*, *escalations=None*, *hostgroups=None*, *checkmodulations=None*, *macromodulations=None*)
Create link between objects:

```
* hosts -> timeperiods
* hosts -> hosts (parents, etc)
* hosts -> commands (check_command)
* hosts -> contacts
```

Parameters

- **timeperiods** (alignak.objects.timeperiod.Timeperiods) – timeperiods to link
- **commands** (alignak.objects.command.Commands) – commands to link
- **contacts** (alignak.objects.contact.Contacts) – contacts to link
- **realms** (alignak.objects.realm.Realms) – realms to link
- **resultmodulations** (alignak.objects.resultmodulation.Resultmodulations) – resultmodulations to link
- **businessimpactmodulations** (alignak.objects.businessimpactmodulation.Businessimpactmodulations) – businessimpactmodulations to link
- **escalations** (alignak.objects.escalation.Escalations) – escalations to link
- **hostgroups** (alignak.objects.hostgroup.Hostgroups) – hostgroups to link
- **checkmodulations** (*alignak.objects.checkmodulation.Checkmodulations*) – checkmodulations to link
- **macromodulations** (*alignak.objects.macromodulation.Macromodulations*) – macromodulations to link

Returns None

linkify_h_by_h()
Link hosts with their parents

Returns None

linkify_h_by_hg(*hostgroups*)
Link hosts with hostgroups

Parameters **hostgroups** (alignak.objects.hostgroup.Hostgroups) – hostgroups object to link with

Returns None

linkify_h_by_realms(*realms*)
Link hosts with realms

Parameters **realms** (alignak.objects.realm.Realms) – realms object to link with

Returns None

name_property = 'host_name'

alignak.objects.hostdependency module

This module provides Hostdependency and Hostdependencies classes that implements dependencies between hosts. Basically used for parsing.

class alignak.objects.hostdependency.**Hostdependencies**(*items*, *index_items=True*, *parsing=True*)
Bases: *alignak.objects.item.Items*

Hostdependencies manage a list of Hostdependency objects, used for parsing configuration

delete_hostsdep_by_id(*ids*)
Delete a list of hostdependency

Parameters **ids** (*list*) – ids list to delete

Returns None

explode(*hostgroups*)
Explode all host dependency for each member of hostgroups Each member of dependent hostgroup or hostgroup in dependency have to get a copy of host dependencies (quite complex to parse)

Parameters **hostgroups** (alignak.objects.hostgroup.Hostgroups) – used to look for hostgroup

Returns None

inner_class
alias of *Hostdependency*

is_correct()
Check if this object configuration is correct

```
* Check our own specific properties
* Call our parent class is_correct checker
```

Returns True if the configuration is correct, otherwise False

Return type bool

linkify(*hosts*, *timeperiods*)
Create link between objects:

```
* hostdependency -> host
* hostdependency -> timeperiods
```

Parameters

- **hosts** (alignak.objects.host.Hosts) – hosts to link
- **timeperiods** (alignak.objects.timeperiod.Timeperiods) – timeperiods to link

Returns None

linkify_h_by_hd(*hosts*)
Add dependency in host objects :param hosts: hosts list :type hosts: alignak.objects.host.Hosts

Returns None

linkify_hd_by_h(*hosts*)
Replace dependent_host_name and host_name in host dependency by the real object

Parameters **hosts** (alignak.objects.host.Hosts) – host list, used to look for a specific one

Returns None

linkify_hd_by_tp(*timeperiods*)
Replace dependency_period by a real object in host dependency

Parameters **timeperiods** (alignak.objects.timeperiod.Timeperiods) – list of timeperiod, used to look for a specific one

Returns None

class alignak.objects.hostdependency.**Hostdependency**(*params=None*, *parsing=True*)
Bases: *alignak.objects.item.Item*

Hostdependency class is a simple implementation of host dependency as defined in a monitoring context (dependency period, notification_failure_criteria ..)

get_name()
Get name based on dependent_host_name and host_name attributes Each attribute is substituted by 'unknown' if attribute does not exist

Returns dependent_host_name/host_name

Return type str

my_type = 'hostdependency'

properties = {'definition_order': <Property <class 'alignak.property.IntegerProp'>, d

alignak.objects.hostescalation module

This module provides Hostescalation and Hostescalations classes that implements host escalation for notification. Basically used for parsing.

class alignak.objects.hostescalation.**Hostescalation**(*params=None*, *parsing=True*)
Bases: *alignak.objects.item.Item*

Hostescalation class is used to implement notification escalation for hosts

TODO: Why this class does not inherit from alignak.objects.Escalation. Maybe we can merge it

my_type = 'hostescalation'

properties = {'contact_groups': <Property <class 'alignak.property.ListProp'>, defaul

class alignak.objects.hostescalation.**Hostescalations**(*items*, *index_items=True*, *parsing=True*)

Bases: *alignak.objects.item.Items*

Hostescalations manage a list of Hostescalation objects, used for parsing configuration

explode(*escalations*)

Create instance of Escalation for each HostEscalation object

Parameters **escalations** (alignak.objects.escalation.Escalations) – list of escalation, used to add new ones

Returns None

inner_class

alias of *Hostescalation*

name_property = ''

alignak.objects.hostextinfo module

This is the main class for the Host ext info. In fact it's mainly about the configuration part. Parameters are merged in Hosts so it's no use in running part

class alignak.objects.hostextinfo.**HostExtInfo**(*params=None*, *parsing=True*)

Bases: *alignak.objects.genericextinfo.GenericExtInfo*

HostExtInfo class is made to handle some parameters of SchedulingItem:

```
* notes
* notes_url
* icon_image
* icon_image_alt
```

TODO: Is this class really necessary?

definition_order

host_name

icon_image

icon_image_alt

imported_from

macros = {'HOSTNAME': 'host_name', 'HOSTNOTES': 'notes', 'HOSTNOTESURL': 'notes_url'}

my_type = 'hostextinfo'

name

notes

notes_url

properties = {'2d_coords': <Property <class 'alignak.property.StringProp'>, default:

register

statusmap_image

use

vrml_image

class alignak.objects.hostextinfo.**HostsExtInfo**(*items*, *index_items=True*, *parsing=True*)

Bases: *alignak.objects.item.Items*

HostsExtInfo manage HostExtInfo and propagate properties (listed before) into Hosts if necessary

inner_class
alias of *HostExtInfo*

merge(*hosts*)
Merge extended host information into services

Parameters **hosts** (alignak.objects.host.Hosts) – hosts list, to look for a specific one

Returns None

static merge_extinfo(*host*, *extinfo*)
Merge extended host information into a host

Parameters

- **host** (alignak.objects.host.Host) – the host to edit
- **extinfo** (alignak.objects.hostextinfo.HostExtInfo) – the external info we get data from

Returns None

name_property = 'host_name'

alignak.objects.hostgroup module

This module provide Hostgroup and Hostgroups class used to manage host groups

class alignak.objects.hostgroup.**Hostgroup**(*params=None*, *parsing=True*)

Bases: *alignak.objects.itemgroup.Itemgroup*

Class to manage a group of host A Hostgroup is used to manage a group of hosts

get_hostgroup_members()
Get the groups members of the group

Returns list of hosts

Return type list

get_hosts()
Get the hosts of the group

Returns list of hosts

Return type list

get_hosts_by_explosion(*hostgroups*)
Get hosts of this group

Parameters **hostgroups** (alignak.objects.hostgroup.Hostgroups) – Hostgroup object

Returns list of hosts of this group

Return type list

get_name()
Get the group name

group_members_property = 'hostgroup_members'

macros = {'HOSTGROUPACTIONURL': 'action_url', 'HOSTGROUPALIAS': 'alias', 'HOSTGROUPGRO

members_property = 'members'

my_type = 'hostgroup'

properties = {'action_url': <Property <class 'alignak.property.StringProp'>, default:

running_properties = {'conf_is_correct': <Property <class 'alignak.property.BoolProp'

class alignak.objects.hostgroup.**Hostgroups**(*items*, *index_items=True*, *parsing=True*)
Bases: *alignak.objects.itemgroup.Itemgroups*

Class to manage list of Hostgroup Hostgroups is used to regroup all Hostgroup

add_member(*host_name*, *hostgroup_name*)
Add a host string to a hostgroup member if the host group do not exist, create it

Parameters **host_name** (*str*) – host name

:param hostgroup_name:hostgroup name :type hostgroup_name: str :return: None

explode()
Fill members with hostgroup_members

Returns None

get_members_of_group(*gname*)
Get all members of a group which name is given in parameter

Parameters **gname** (*str*) – name of the group

Returns list of the hosts in the group

Return type list[*alignak.objects.host.Host*]

inner_class
alias of *Hostgroup*

linkify(*hosts=None*, *realms=None*, *forced_realms_hostgroups=True*)
Link hostgroups with hosts and realms

Parameters

- **hosts** (alignak.objects.host.Hosts) – all Hosts
- **realms** (alignak.objects.realm.Realms) – all Realms

Returns None

linkify_hostgroups_hosts(*hosts*)
We just search for each hostgroup the id of the hosts and replace the names by the found identifiers

Parameters **hosts** (alignak.objects.host.Hosts) – object Hosts

Returns None

linkify_hostgroups_realms_hosts(*realms*, *hosts*, *forced_realms_hostgroups=True*)
Link between an hostgroup and a realm is already done in the configuration parsing function that defines and checks the default satellites, realms, hosts and hosts groups consistency.

This function will only raise some alerts if hosts groups and hosts that are contained do not belong the same realm !

Parameters

- **realms** (alignak.objects.realm.Realms) – object Realms
- **hosts** (alignak.objects.host.Hosts) – object Realms

Returns None

name_property = 'hostgroup_name'

alignak.objects.item module

This class is a base class for nearly all configuration elements like service, hosts or contacts.

class alignak.objects.item.**Item**(*params=None*, *parsing=True*)

Bases: *alignak.alignakobject.AlignakObject*

Class to manage an item

An Item is the base of many objects of Alignak. So it defines properties that are common to all the objects: - name - register: it is a real object (True) or a template definition (False) - imported_from: source configuration file or backend - use: templates which this object inherits from - definition_order: priority if the same object is defined several times - tags: the information tags attached to an object

Note: some tags are automatically set on an object when it uses some templates.

And some configuration parsing information: - conf_is_correct: whether configuration is correct or not - configuration_warnings: list of configuration parsing warning log - configuration_errors: list of configuration parsing error log

common functions.

add_comment(*comment*)

Add a comment to this object

Parameters **comment** (*object*) – a Comment object

Returns None

add_downtime(*downtime*)

Add a downtime in this object

Parameters **downtime** (*object*) – a Downtime object

Returns None

add_error(*txt*)

Add a message in the configuration errors list so we can print them all in one place

Set the object configuration as not correct

Parameters **txt** (*str*) – error message

Returns None

add_warning(*txt*)

Add a message in the configuration warnings list so we can print them all in one place

Parameters **txt** (*str*) – warning message

Returns None

clean()
Clean properties only needed for initialization and configuration

Returns None

copy()
Get a copy of this item but with a new id

Returns copy of this object with a new id

Return type object

del_comment(*comment_id*)
Delete a comment in this object

Parameters **comment_id** (*int*) – id of the comment to delete

Returns None

del_downtime(*downtime_id*)
Delete a downtime in this object

Parameters **downtime_id** (*int*) – id of the downtime to delete

Returns None

dump(*dump_file_name=None*)
Dump Item object properties

Returns dictionary with properties

Return type dict

fill_data_brok_from(*data*, *brok_type*)
Add properties to 'data' parameter with properties of this object when 'brok_type' parameter is defined in fill_brok of these properties

Parameters

- **data** (*object*) – object to fill
- **brok_type** (*var*) – name of brok_type

Returns None

get_all_plus_and_delete()
Get all self.plus items of list. We copy it, delete the original and return the copy list

Returns list of self.plus

Return type list

get_check_result_brok()
Create check_result brok

Returns Brok object

Return type alignak.Brok

get_full_name()
Accessor to name attribute

Returns name

Return type str

get_initial_status_brok(*extra=None*)
Create an initial status brok

Parameters **extra** (*dict*) – some extra information to be added in the brok data

Returns Brok object

Return type alignak.Brok

get_name()
Get the name of the item

TODO: never called anywhere, still useful?

Returns the object name string

Return type str

get_new_brok(*name*)
Create a new item brok

Parameters **name** (*str*) – name of the new object

Returns Brok object

Return type alignak.Brok

get_next_schedule_brok()
Create next_schedule (next check) brok

Returns Brok object

Return type alignak.Brok

get_plus_and_delete(*prop*)
get a copy of the property (parameter) in self.plus, delete the original and return the value of copy

Parameters **prop** (*str*) – a property

Returns return the value of the property

Return type str

get_property_value_for_brok(*prop*, *tab*)
Get the property of an object and brok_transformation if needed and return the value

Parameters

- **prop** (*str*) – property name
- **tab** (*object*) – object with all properties of an object

Returns value of the property original or brok converted

Return type str

get_raw_import_values()
Get properties => values of this object

TODO: never called anywhere, still useful?

Returns dictionary of properties => values

Return type dict

get_snapshot_brok(*snap_output*, *exit_status*)
Create snapshot (check_result type) brok

Parameters

- **snap_output** (*str*) – value of output

- **exit_status** (*integer*) – status of exit

Returns Brok object

Return type alignak.Brok

get_templates()
Get list of templates this object use

Returns list of templates

Return type list

get_update_status_brok()
Create an update item brok

Returns Brok object

Return type alignak.Brok

has_plus(*prop*)
Check if self.plus list have this property

Parameters **prop** (*str*) – property to check

Returns True is self.plus has this property, otherwise False

Return type bool

init_running_properties()
Initialize the running_properties. Each instance have own property.

Returns None

is_correct()
Check if this object is correct

This function: - checks if the required properties are defined, ignoring special_properties if some exist - logs the previously found warnings and errors

Returns True if it's correct, otherwise False

Return type bool

is_tpl()
Check if this object is a template

Returns True if is a template, else False

Return type bool

classmethod load_global_conf(*global_configuration*)
Apply global Alignak configuration.

Some objects inherit some properties from the global configuration if they do not define their own value. E.g. the global 'accept_passive_service_checks' is inherited by the services as 'accept_passive_checks'

Parameters

- **cls** (*object*) – parent object
- **global_configuration** (*object*) – current object (child)

Returns None

macros = {}

my_type = ''

ok_up = ''

old_properties_names_to_new()
This function is used by service and hosts to transform Nagios2 parameters to Nagios3 ones, like normal_check_interval to check_interval. There is a old_parameters tab in Classes that give such modifications to do.

Returns None

properties = {'definition_order': <Property <class 'alignak.property.IntegerProp'>, d

running_properties = {'conf_is_correct': <Property <class 'alignak.property.BoolProp'

serialize()
This function serialize into a simple dict object. It is used when transferring data to other daemons over the network (http)

Here is the generic function that simply export attributes declared in the properties dictionary and the running_properties of the object.

Returns Dictionary containing key and value from properties and running_properties

Return type dict

class alignak.objects.item.**Items**(*items*, *index_items=True*, *parsing=True*)
Bases: object

Class to manage all Item

add_error(*txt*)

Add a message in the configuration errors list so we can print them all in one place

Set the object configuration as not correct

Parameters **txt** (*str*) – error message

Returns None

add_item(*item*, *index=True*)
Add an item into our containers, and index it depending on the *index* flag.

Parameters

- **item** (alignak.objects.item.Item) – object to add
- **index** (*bool*) – Flag indicating if the item should be indexed

Returns the new items created

:rtype list

add_items(*items*, *index_items*)
Add items to template if is template, else add in item list

Parameters

- **items** (alignak.objects.item.Items) – items list to add
- **index_items** (*bool*) – Flag indicating if the items should be indexed on the fly.

Returns None

add_template(*tpl*)
Add and index a template into the *templates* container.

Parameters **tpl** (alignak.objects.item.Item) – The template to add

Returns None

add_warning(*txt*)

Add a message in the configuration warnings list so we can print them all in one place

Parameters **txt** (*str*) – warning message

Returns None

apply_inheritance()
For all items and templates inherit properties and custom variables.

Returns None

apply_partial_inheritance(*prop*)
Define property with inheritance value of the property

Parameters **prop** (*str*) – property

Returns None

clean()
Clean the list items

Returns None

static evaluate_hostgroup_expression(*expr*, *hosts*, *hostgroups*, *look_in='hostgroups'*)
Evaluate hostgroup expression

Parameters

- **expr** (*str*) – an expression
- **hosts** (alignak.objects.host.Hosts) – hosts object (all hosts)
- **hostgroups** (alignak.objects.hostgroup.Hostgroups) – hostgroups object (all hostgroups)
- **look_in** (*str*) – item name where search

Returns return list of hostgroups

Return type list

static explode_contact_groups_into_contacts(*item*, *contactgroups*)
Get all contacts of contact_groups and put them in contacts container

Parameters

- **item** (*object*) – item where have contact_groups property
- **contactgroups** (alignak.objects.contactgroup.Contactgroups) – all contactgroups object

Returns None

explode_host_groups_into_hosts(*item*, *hosts*, *hostgroups*)
Get all hosts of hostgroups and add all in host_name container

Parameters

- **item** (alignak.objects.item.Item) – the item object
- **hosts** (alignak.objects.host.Hosts) – hosts object

- **hostgroups** (alignak.objects.hostgroup.Hostgroups) – hostgroups object

Returns None

fill_default()
Define properties for each items with default value when not defined

Returns None

find_by_name(*name*)
Find an item by name

Parameters **name** (*str*) – name of item

Returns item

Return type *alignak.objects.item.Item*

find_tpl_by_name(*name*)
Find template by name

Parameters **name** (*str*) – name of template

Returns name of template found

Return type str | None

get_all_tags(*item*)
Get all tags of an item

Parameters **item** (Item) – an item

Returns list of tags

Return type list

get_customs_properties_by_inheritance(*obj*)
Get custom properties from the templates defined in this object

Parameters **obj** (alignak.objects.item.Item) – the oject to search the property

Returns list of custom properties

Return type list

static get_hosts_from_hostgroups(*hgname*, *hostgroups*)
Get hosts of hostgroups

Parameters

- **hgname** (*str*) – hostgroup name
- **hostgroups** (alignak.objects.hostgroup.Hostgroups) – hostgroups object (all hostgroups)

Returns list of hosts

Return type list

get_property_by_inheritance(*obj*, *prop*)
Get the property asked in parameter to this object or from defined templates of this object

todo: rewrite this function which is really too complex!

Parameters

- **obj** (alignak.objects.item.Item) – the object to search the property

- **prop** (*str*) – name of property

Returns Value of property of this object or of a template

Return type str or None

index_item(*item*)
Index an item into our *name_to_item* dictionary. If an object holding the same item's name/key already exists in the index then the conflict is managed by the *manage_conflict* method.

Parameters **item** (alignak.objects.item.Item) – item to index

Returns item modified

Return type object

index_template(*tpl*)
Indexes a template by *name* into the *name_to_template* dictionary.

Parameters **tpl** (alignak.objects.item.Item) – The template to index

Returns None

inner_class
alias of *Item*

is_correct()
Check if the items list configuration is correct

```
* check if duplicate items exist in the list and warn about this
* set alias and display_name property for each item in the list if they do␣
↪not exist
* check each item in the list
* log all previous warnings
* log all previous errors
```

Returns True if the configuration is correct, otherwise False

Return type bool

linkify_item_templates(*item*)
Link templates

Parameters **item** (alignak.objects.item.Item) – an item

Returns None

linkify_s_by_module(*modules*)
Link modules to items

Parameters **modules** (alignak.objects.module.Modules) – Modules object (list of all the modules found in the configuration)

Returns None

linkify_templates()
Link all templates, and create the template graph too

Returns None

linkify_with_business_impact_modulations(*business_impact_modulations*)
Link items with business impact objects

Parameters **business_impact_modulations** (*alignak.objects.businessmodulation.Businessmodulations*) – all business impacts object

Returns None

linkify_with_checkmodulations(*checkmodulations*)
Link checkmodulation object

Parameters **checkmodulations** (*alignak.objects.checkmodulation.Checkmodulations*) – checkmodulations object

Returns None

linkify_with_contacts(*contacts*)
Link items with contacts items

Parameters **contacts** (alignak.objects.contact.Contacts) – all contacts object

Returns None

linkify_with_escalations(*escalations*)
Link with escalations

Parameters **escalations** (alignak.objects.escalation.Escalations) – all escalations object

Returns None

linkify_with_macromodulations(*macromodulations*)
Link macromodulations

Parameters **macromodulations** (*alignak.objects.macromodulation.Macromodulations*) – macromodulations object

Returns None

linkify_with_resultmodulations(*resultmodulations*)
Link items with resultmodulations items

Parameters **resultmodulations** (*alignak.resultmodulation.Resultmodulations*) – all resultmodulations object

Returns None

linkify_with_timeperiods(*timeperiods*, *prop*)
Link items with timeperiods items

Parameters

- **timeperiods** (alignak.objects.timeperiod.Timeperiods) – all timeperiods object
- **prop** (*str*) – property name

Returns None

manage_conflict(*item*, *name*)
Checks if an object holding the same name already exists in the index.

If so, it compares their definition order: the lowest definition order is kept. If definition order equal, an error is risen.Item

The method returns the item that should be added after it has decided which one should be kept.

If the new item has precedence over the New existing one, the existing is removed for the new to replace it.

Parameters

- **item** (alignak.objects.item.Item) – object to check for conflict
- **name** (*str*) – name of the object

Returns 'item' parameter modified

Return type object

no_loop_in_parents(*attr1*, *attr2*)

Find loop in dependencies. For now, used with the following attributes : :(self, parents):

host dependencies from host object

(host_name, dependent_host_name) host dependencies from hostdependencies object

(service_description, dependent_service_description) service dependencies from servicedependencies object

Parameters

- **attr1** (*str*) – attribute name
- **attr2** (*str*) – attribute name

Returns list

Return type list

old_properties_names_to_new()

Convert old Nagios2 names to Nagios3 new names

TODO: still useful?

Returns None

remove_item(*item*)

Remove (and un-index) an object

Parameters **item** (alignak.objects.item.Item) – object to remove

Returns None

remove_template(*tpl*)

Removes and un-index a template from the *templates* container.

Parameters **tpl** (alignak.objects.item.Item) – The template to remove

Returns None

remove_templates()

Remove templates

Returns None

serialize()

This function serialize items into a simple dict object. It is used when transferring data to other daemons over the network (http)

Here is the generic function that simply serialize each item of the items object

Returns Dictionary containing item's uuid as key and item as value

Return type dict

unindex_item(*item*)
Un-index an item from our name_to_item dict. :param item: the item to un-index :type item: alignak.objects.item.Item :return: None

unindex_template(*tpl*)
Unindex a template from the *templates* container.

Parameters **tpl** (alignak.objects.item.Item) – The template to un-index

Returns None

alignak.objects.itemgroup module

This module provide Itemgroup and Itemgroups class used to define group of items

class alignak.objects.itemgroup.**Itemgroup**(*params=None*, *parsing=True*)
Bases: *alignak.objects.item.Item*

Class to manage a group of items An Itemgroup is used to group items (eg. Host, Service,...)

add_members(*members*)
Add a new member to the members list

Parameters **members** (*str*) – member name

Returns None

add_unknown_members(*members*)
Add a new member to the unknown members list

Parameters **member** (*str*) – member name

Returns None

copy_shell()
Copy the group properties EXCEPT the members. Members need to be filled after manually

Returns Itemgroup object

Return type *alignak.objects.itemgroup.Itemgroup*

Returns None

get_initial_status_brok(*extra=None*)
Get a brok with the group properties

members contains a list of uuid which we must provide the names. Thus we will replace the default provided uuid with the members short name. The *extra* parameter, if present,

is containing the Items to search for...

Parameters **extra** (alignak.objects.item.Items) – monitoring items, used to recover members

:return:Brok object :rtype: object

get_members()
Get the members of the group

Returns list of members

Return type list

group_members_property = ''

is_correct()
Check if a group is valid. Valid mean all members exists, so list of unknown_members is empty

Returns True if group is correct, otherwise False

Return type bool

members_property = 'members'

properties = {'definition_order': <Property <class 'alignak.property.IntegerProp'>, d

replace_members(*members*)
Replace members of itemgroup by new members list

Parameters **members** (*list*) – list of members

Returns None

running_properties = {'conf_is_correct': <Property <class 'alignak.property.BoolProp'

class alignak.objects.itemgroup.**Itemgroups**(*items*, *index_items=True*, *parsing=True*)
Bases: *alignak.objects.item.Items*

Class to manage list of groups of items An itemgroup is used to regroup items group

add(*itemgroup*)
Add an item (itemgroup) to Itemgroups

Parameters **itemgroup** (alignak.objects.itemgroup.Itemgroup) – an item

Returns None

alignak.objects.macromodulation module

This module provide MacroModulation and MacroModulations classes used to change critical and warning level in some periods (like the night)

class alignak.objects.macromodulation.**MacroModulation**(*params=None*, *parsing=True*)
Bases: *alignak.objects.item.Item*

Class to manage a MacroModulation A MacroModulation is defined to change critical and warning level in some periods (like the night)

get_name()
Get the name of the macromodulation

Returns the macromodulation name string

Return type str

is_active(*timperiods*)
Know if this macro is active for this correct period

Returns True is we are in the period, otherwise False

Return type bool

is_correct()
Check if this object configuration is correct

```
* Call our parent class is_correct checker
```

Returns True if the configuration is correct, otherwise False

Return type bool

macros = {}

my_type = 'macromodulation'

properties = {'definition_order': <Property <class 'alignak.property.IntegerProp'>, d

running_properties = {'conf_is_correct': <Property <class 'alignak.property.BoolProp'

special_properties = ('modulation_period',)

class alignak.objects.macromodulation.**MacroModulations**(*items*, *index_items=True*, *parsing=True*)

Bases: *alignak.objects.item.Items*

Class to manage all MacroModulation

inner_class
alias of *MacroModulation*

linkify(*timeperiods*)
Link with timeperiod

Parameters **timeperiods** (alignak.objects.timeperiod.Timeperiods) – Timeperiod object

Returns None

name_property = 'macromodulation_name'

alignak.objects.module module

This module provide Module and Modules classes used to manage internal and external modules for each daemon

class alignak.objects.module.**Module**(*params=None*, *parsing=True*)

Bases: *alignak.objects.item.Item*

Class to manage a module

get_name()
Get name of module

Returns Name of module

Return type str

get_types()
Get types of the module

Returns Types of the module

Return type str

is_a_module(*module_type*)
Is the module of the required type?

Parameters **module_type** – module type to check

Type str

Returns True / False

macros = {}

my_type = 'module'

properties = {'daemon': <Property <class 'alignak.property.StringProp'>, default: u'

serialize()
A module may have some properties that are not defined in the class properties list. Serializing a module is the same as serializing an Item but we also also include all the existing properties that are not defined in the properties or running_properties class list.

We must also exclude the reference to the daemon that loaded the module!

class alignak.objects.module.**Modules**(*items*, *index_items=True*, *parsing=True*)
Bases: *alignak.objects.item.Items*

Class to manage list of modules Modules is used to group all Module

inner_class
alias of *Module*

linkify()
Link a module to some other modules

Returns None

linkify_s_by_plug()
Link a module to some other modules

Returns None

name_property = 'name'

alignak.objects.notificationway module

This module provides NotificationWay and NotificationWays classes that implements way of sending notifications. Basically used for parsing.

class alignak.objects.notificationway.**NotificationWay**(*params=None*, *parsing=True*)
Bases: *alignak.objects.item.Item*

NotificationWay class is used to implement way of sending notifications (command, periods..)

get_name()
Accessor to notificationway_name attribute

Returns notificationway name

Return type str

get_notification_commands(*o_type*)
Get notification commands for object type

Parameters **o_type** (*str*) – object type (host or service)

Returns command list

Return type list[*alignak.objects.command.Command*]

is_correct()
Check if this object configuration is correct

```
* Check our own specific properties
* Call our parent class is_correct checker
```

Returns True if the configuration is correct, otherwise False

Return type bool

macros = {}

my_type = 'notificationway'

old_properties = {'min_criticity': 'min_business_impact'}

properties = {'definition_order': <Property <class 'alignak.property.IntegerProp'>, d

running_properties = {'conf_is_correct': <Property <class 'alignak.property.BoolProp'

serialize()

This function serialize into a simple dict object. It is used when transferring data to other daemons over the network (http)

Here is the generic function that simply export attributes declared in the properties dictionary and the running_properties of the object.

Returns Dictionary containing key and value from properties and running_properties

Return type dict

special_properties = ('service_notification_commands', 'host_notification_commands', '

want_host_notification(*timperiods*, *timestamp*, *state*, *n_type*, *business_impact*, *cmd=None*)

Check if notification options match the state of the host Notification is NOT wanted in ONE of the following case:

```
* host notifications are disabled
* cmd is not in host_notification_commands
* business_impact < self.min_business_impact
* host_notification_period is not valid
* state does not match host_notification_options for problem, recovery,␣
↪flapping and dt
```

Parameters

- **timestamp** (*int*) – time we want to notify the contact (usually now)
- **state** (*str*) – host or service state (“WARNING”, “CRITICAL” ..)
- **n_type** (*str*) – type of notification (“PROBLEM”, “RECOVERY” ..)
- **business_impact** (*int*) – impact of this service
- **cmd** (*str*) – command launched to notify the contact

Returns True if no condition is matched, otherwise False

Return type bool

TODO: Simplify function

want_service_notification(*timeperiods*, *timestamp*, *state*, *n_type*, *business_impact*, *cmd=None*)

Check if notification options match the state of the service Notification is NOT wanted in ONE of the following case:

```
* service notifications are disabled
* cmd is not in service_notification_commands
* business_impact < self.min_business_impact
* service_notification_period is not valid
* state does not match service_notification_options for problem, recovery␣
↪and flapping
* state does not match host_notification_options for downtime
```

Parameters

- **timestamp** (*int*) – time we want to notify the contact (usually now)
- **state** (*str*) – host or service state ("WARNING", "CRITICAL" ..)
- **n_type** (*str*) – type of notification ("PROBLEM", "RECOVERY" ..)
- **business_impact** (*int*) – impact of this service
- **cmd** (*str*) – command launched to notify the contact

Returns True if no condition is matched, otherwise False

Return type bool

TODO: Simplify function

class alignak.objects.notificationway.**NotificationWays**(*items*, *index_items=True*, *parsing=True*)

Bases: *alignak.objects.commandcallitem.CommandCallItems*

NotificationWays manage a list of NotificationWay objects, used for parsing configuration

inner_class
alias of *NotificationWay*

linkify(*timeperiods*, *commands*)
Create link between objects:

```
* notificationways -> timeperiods
* notificationways -> commands
```

Parameters

- **timeperiods** (alignak.objects.timeperiod.Timeperiods) – timeperiods to link
- **commands** (alignak.objects.command.Commands) – commands to link

Returns None

name_property = 'notificationway_name'

new_inner_member(*name*, *params*)
Create new instance of NotificationWay with given name and parameters and add it to the item list

Parameters

- **name** (*str*) – notification way name
- **params** (*dict*) – notification wat parameters

Returns None

alignak.objects.pollerlink module

This module provide PollerLink and PollerLinks classes used to manage link between the modules Arbiter and Poller

class alignak.objects.pollerlink.**PollerLink**(*params=None*, *parsing=True*)
Bases: *alignak.objects.satellitelink.SatelliteLink*

Class to manage the link between Arbiter and Poller. With this, an arbiter can communicate with a poller

my_type = 'poller'

properties = {'active': <Property <class 'alignak.property.BoolProp'>, default: True

class alignak.objects.pollerlink.**PollerLinks**(*items*, *index_items=True*, *parsing=True*)
Bases: *alignak.objects.satellitelink.SatelliteLinks*

Class to manage list of PollerLink. PollerLinks is used to regroup all links between the Arbiter and different Pollers

inner_class
alias of *PollerLink*

name_property = 'poller_name'

alignak.objects.reactionnerlink module

This module provide ReactionnerLink and ReactionnerLinks classes used to manage reactionners

class alignak.objects.reactionnerlink.**ReactionnerLink**(*params=None*, *parsing=True*)
Bases: *alignak.objects.satellitelink.SatelliteLink*

Class to manage the reactionner information

my_type = 'reactionner'

properties = {'active': <Property <class 'alignak.property.BoolProp'>, default: True

class alignak.objects.reactionnerlink.**ReactionnerLinks**(*items*, *index_items=True*, *parsing=True*)
Bases: *alignak.objects.satellitelink.SatelliteLinks*

Class to manage list of ReactionnerLink. ReactionnerLinks is used to regroup all reactionners

inner_class
alias of *ReactionnerLink*

name_property = 'reactionner_name'

alignak.objects.realm module

This module provides Realm and Realms classes that implements realm for Alignak. Basically used for parsing.

class alignak.objects.realm.**Realm**(*params=None*, *parsing=True*)
Bases: *alignak.objects.itemgroup.Itemgroup*

Realm class is used to implement realm. It is basically a group of Hosts assigned to a specific Scheduler/Poller (other daemon are optional)

add_group_members(*members*)
Add a new group member to the groups list

Parameters **members** (*str*) – member name

Returns None

get_all_subs_satellites_by_type(*sat_type*, *realms*)
Get all satellites of the wanted type in this realm recursively

Parameters

- **sat_type** – satellite type wanted (scheduler, poller ..)
- **realms** (*list of realm object*) – all realms

Returns list of satellite in this realm

Return type list

get_links_for_a_broker(*pollers*, *reactionners*, *receivers*, *realms*, *manage_sub_realms=False*)
Get a configuration dictionary with pollers, reactionners and receivers links for a broker

Parameters

- **pollers** – pollers
- **reactionners** – reactionners
- **receivers** – receivers
- **realms** – realms
- **manage_sub_realms** (*True if the borker manages sub realms*) –

Returns dict containing pollers, reactionners and receivers links (key is satellite id)

Return type dict

get_links_for_a_scheduler(*pollers*, *reactionners*, *brokers*)
Get a configuration dictionary with pollers, reactionners and brokers links for a scheduler

Returns dict containing pollers, reactionners and brokers links (key is satellite id)

Return type dict

get_name()
Accessor to realm_name attribute

Returns realm name

Return type str

get_nb_of_must_have_satellites(*s_type*)
Generic function to access one of the number satellite attribute ie : self.nb_pollers, self.nb_reactionners ...

Parameters **s_type** (*str*) – satellite type wanted

Returns self.nb_*type*s

Return type int

get_potential_satellites_by_type(*satellites*, *s_type*)
Generic function to access one of the potential satellite attribute ie : self.potential_pollers, self.potential_reactionners ...

Parameters

- **satellites** (*SatelliteLink list*) – list of SatelliteLink objects
- **s_type** (*str*) – satellite type wanted

Returns self.potential_*type*s

Return type list

get_realms_by_explosion(*realms*)
Get all members of this realm including members of sub-realms on multi-levels

Parameters **realms** (alignak.objects.realm.Realms) – realms list, used to look for a specific one

Returns list of members and add realm to realm_members attribute

Return type list

get_satellites_by_type(*s_type*)
Generic function to access one of the satellite attribute ie : self.pollers, self.reactionners ...

Parameters **s_type** (*str*) – satellite type wanted

Returns self.*type*s

Return type list

group_members_property = 'realm_members'

macros = {'REALMDEFAULT': 'default', 'REALMGROUP_MEMBERS': 'group_members', 'REALMHOST

members_property = 'members'

my_type = 'realm'

name
Get the realm name

prepare_satellites(*satellites*)
Update the following attributes of a realm:

```
* nb_*satellite type*s
* self.potential_*satellite type*s
```

(satellite types are scheduler, reactionner, poller, broker and receiver)

Parameters **satellites** (*dict*) – dict of SatelliteLink objects

Returns None

properties = {'alias': <Property <class 'alignak.property.StringProp'>, default: u''

running_properties = {'actively_checked_hosts': <Property <class 'alignak.property.Bo

set_level(*level*, *realms*)
Set the realm level in the realms hierarchy

Returns None

class alignak.objects.realm.**Realms**(*items*, *index_items=True*, *parsing=True*)
Bases: *alignak.objects.itemgroup.Itemgroups*

Realms manage a list of Realm objects, used for parsing configuration

explode()
Explode realms with each realm_members and higher_realms to get all the realms sub realms.

Returns None

get_default(*check=False*)
Get the default realm

Parameters **check** (*bool*) – check correctness if True

Returns Default realm of Alignak configuration

Return type alignak.objects.realm.Realm | None

inner_class
alias of *Realm*

linkify()
The realms linkify is done during the default realms/satellites initialization in the Config class.

This functione only finishes the process by setting the realm level property according to the realm position in the hierarchy.

All ' level' 0 realms are main realms that have their own hierarchy.

Returns None

name_property = 'realm_name'

prepare_satellites(*satellites*)
Init the following attributes for each realm:

```
* to_satellites (with *satellite type* keys)
* to_satellites_need_dispatch (with *satellite type* keys)
* to_satellites_managed_by (with *satellite type* keys)
* nb_*satellite type*s
* self.potential_*satellite type*s
```

(satellite type are reactionner, poller, broker and receiver)

Parameters **satellites** (*alignak.object.satelittelink.SatelliteLink*) – (schedulers, brokers, reactionners, pollers, receivers)

Returns None

alignak.objects.receiverlink module

This module provide ReceiverLink and ReceiverLinks classes used to manage receivers

class alignak.objects.receiverlink.**ReceiverLink**(*params=None*, *parsing=True*)
Bases: *alignak.objects.satellitelink.SatelliteLink*

Class to manage the receiver information

my_type = 'receiver'

properties = {'active': <Property <class 'alignak.property.BoolProp'>, default: True

class alignak.objects.receiverlink.**ReceiverLinks**(*items*, *index_items=True*, *parsing=True*)
Bases: *alignak.objects.satellitelink.SatelliteLinks*

Class to manage list of ReceiverLink. ReceiverLinks is used to regroup all receivers

inner_class
alias of *ReceiverLink*

name_property = 'receiver_name'

alignak.objects.resultmodulation module

This module provide Resultmodulation and Resultmodulations classes used to describe the modulation of a check command. Modulation occurs on a modulation period (Timeperiod)

class alignak.objects.resultmodulation.**Resultmodulation**(*params=None*, *parsing=True*)

Bases: *alignak.objects.item.Item*

Resultmodulation class is simply a modulation of a check result exit code during a modulation_period.

get_name()

Accessor to resultmodulation_name attribute

Returns result modulation name

Return type str

is_active(*timperiods*)

Know if this result modulation is active now

Returns True is we are in the period, otherwise False

Return type bool

module_return(*return_code*, *timeperiods*)

Module the exit code if necessary

```
* modulation_period is legit
* exit_code_modulation
* return_code in exit_codes_match
```

Parameters **return_code** (*int*) – actual code returned by the check

Returns return_code modulated if necessary (exit_code_modulation)

Return type int

my_type = 'resultmodulation'

properties = {'definition_order': <Property <class 'alignak.property.IntegerProp'>, d

special_properties = ('modulation_period',)

class alignak.objects.resultmodulation.**Resultmodulations**(*items*, *index_items=True*, *parsing=True*)

Bases: *alignak.objects.item.Items*

Resultmodulations class allowed to handle easily several CheckModulation objects

inner_class

alias of *Resultmodulation*

linkify(*timeperiods*)

Wrapper for linkify_rm_by_tp Replace check_period by real Timeperiod object into each Resultmodulation

Parameters **timeperiods** (alignak.objects.timeperiod.Timeperiods) – timeperiods to link to

Returns None

name_property = 'resultmodulation_name'

alignak.objects.satellitelink module

This module provides an abstraction layer for communications between Alignak daemons Used by the Arbiter

exception alignak.objects.satellitelink.**LinkError**(*msg*)
Bases: exceptions.Exception

Exception raised for errors with the satellite links.

Attributes: msg – explanation of the error

class alignak.objects.satellitelink.**SatelliteLink**(*params=None*, *parsing=True*)
Bases: *alignak.objects.item.Item*

SatelliteLink is a common Class for links between Arbiter and other satellites. Used by the Dispatcher object.

add_failed_check_attempt(*reason=''*)
Set the daemon as unreachable and add a failed attempt if we reach the maximum attempts, set the daemon as dead

Parameters **reason** (*str*) – the reason of adding an attempts (stack trace sometimes)

Returns None

communicate(***outer_kwargs*)
Check if the daemon connection is authorized and valid

create_connection()
Initialize HTTP connection with a satellite (con attribute) and set its uri attribute

This is called on the satellite link initialization

Returns None

static get_a_satellite_link(*sat_type*, *sat_dict*)
Get a SatelliteLink object for a given satellite type and a dictionary

Parameters

- **sat_type** – type of satellite
- **sat_dict** – satellite configuration data

Returns

get_actions(***kwargs*)

get_and_clear_broks()
Get and clean all of our broks

Returns list of all broks of the satellite link

Return type list

get_and_clear_context()
Get and clean all of our broks, actions, external commands and homerun

Returns list of all broks of the satellite link

Return type list

get_broks(***kwargs*)

get_conf(***kwargs*)

get_daemon_stats(***kwargs*)

get_events(***kwargs*)

get_external_commands(***kwargs*)

get_initial_broks(***kwargs*)

get_livestate()
Get the SatelliteLink live state.

The live state is a tuple information containing a state identifier and a message, where: state is: - 0 for an up and running satellite - 1 if the satellite is not reachale - 2 if the satellite is dead - 3 else (not active)

Returns tuple

get_results(***kwargs*)

get_running_id(***kwargs*)

give_satellite_cfg()
Get the default information for a satellite.

Overridden by the specific satellites links

Returns dictionary of information common to all the links

Return type dict

give_satellite_json()
Get the json information for a satellite.

This to provide information that will be exposed by a daemon on its HTTP interface.

Returns dictionary of information common to all the links

Return type dict

has_a_conf(***kwargs*)

manages(*cfg_part*)
Tell if the satellite is managing this configuration part

The managed configuration is formed as a dictionary indexed on the link instance_id:

{

u'SchedulerLink_1': { u'hash': u'4d08630a3483e1eac7898e7a721bd5d7768c8320', u'push_flavor': u'4d08630a3483e1eac7898e7a721bd5d7768c8320', u'managed_conf_id': [u'Config_1']

}

}

Note that the managed configuration is a string array rather than a simple string... no special for this reason, probably due to the serialization when the configuration is pushed :/

Parameters **cfg_part** (*Conf*) – configuration part as prepare by the Dispatcher

Returns True if the satellite manages this configuration

Return type bool

prepare_for_conf()
Initialize the pushed configuration dictionary with the inner properties that are to be propagated to the satellite link.

Returns None

properties = {'active': <Property <class 'alignak.property.BoolProp'>, default: True

push_actions(***kwargs*)

push_broks(***kwargs*)

push_external_commands(***kwargs*)

push_results(***kwargs*)

put_conf(***kwargs*)

running_properties = {'alive': <Property <class 'alignak.property.BoolProp'>, default

scheme
Daemon interface scheme

Returns http or https if the daemon uses SSL

Return type str

set_alive()
Set alive, reachable, and reset attempts. If we change state, raise a status brok update

alive, means the daemon is prenset in the system reachable, means that the HTTP connection is valid

With this function we confirm that the daemon is reachable and, thus, we assume it is alive!

Returns None

set_arbiter_satellite_map(*satellite_map=None*)

satellite_map is the satellites map in current context:

- A SatelliteLink is owned by an Arbiter
- satellite_map attribute of a SatelliteLink is the map defined

IN THE satellite configuration but for creating connections, we need to have the satellites map from the Arbiter point of view

Returns None

set_dead()
Set the satellite into dead state: If we change state, raise a status brok update

:return:None

stop_request(***kwargs*)

update_infos(***kwargs*)

valid_connection(***outer_kwargs*)
Check if the daemon connection is established and valid

wait_new_conf(***kwargs*)

class alignak.objects.satellitelink.**SatelliteLinks**(*items*, *index_items=True*, *parsing=True*)

Bases: *alignak.objects.item.Items*

Class to handle serveral SatelliteLink

inner_class
alias of *SatelliteLink*

linkify(*modules*)
Link modules and Satellite links

Parameters **modules** (alignak.objects.module.Modules) – Module object list

Returns None

name_property = 'name'

alignak.objects.schedulerlink module

This module provide SchedulerLink and SchedulerLinks classes used to manage schedulers

class alignak.objects.schedulerlink.**SchedulerLink**(*params=None*, *parsing=True*)

Bases: *alignak.objects.satellitelink.SatelliteLink*

Class to manage the scheduler information

get_override_configuration()

Some parameters can give as 'overridden parameters' like use_timezone so they will be mixed (in the scheduler) with the standard conf sent by the arbiter

Returns dictionary of properties

Return type dict

my_type = 'scheduler'

properties = {'accept_passive_unknown_check_results': <Property <class 'alignak.prope

running_properties = {'alive': <Property <class 'alignak.property.BoolProp'>, default

class alignak.objects.schedulerlink.**SchedulerLinks**(*items*, *index_items=True*, *parsing=True*)

Bases: *alignak.objects.satellitelink.SatelliteLinks*

Please Add a Docstring to describe the class here

inner_class

alias of *SchedulerLink*

name_property = 'scheduler_name'

alignak.objects.schedulingitem module

This class is a common one for service/host. Here you will find all scheduling related functions, like the schedule or the consume_check. It's a very important class!

class alignak.objects.schedulingitem.**SchedulingItem**(*params=None*, *parsing=True*)

Bases: *alignak.objects.item.Item*

SchedulingItem class provide method for Scheduler to handle Service or Host objects

acknowledge_problem(*notification_period*, *hosts*, *services*, *sticky*, *notify*, *author*, *comment*, *end_time=0*)

Add an acknowledge

Parameters

- **sticky** (*integer*) – acknowledge will be always present is host return in UP state
- **notify** (*integer*) – if to 1, send a notification
- **author** (*str*) – name of the author or the acknowledge
- **comment** (*str*) – comment (description) of the acknowledge

- **end_time** (*int*) – end (timeout) of this acknowledge in seconds(timestamp) (0 to never end)

Returns None | alignak.comment.Comment

acknowledged
Simple property renaming for better API;)

add_attempt()
Add an attempt when a object is a non-ok state

Returns None

add_flapping_change(*sample*)
Add a flapping sample and keep cls.flap_history samples

Parameters **sample** (*bool*) – Sample to add

Returns None

business_rule_notification_is_blocked(*hosts*, *services*)
Process business rule notifications behaviour. If all problems have been acknowledged, no notifications should be sent if state is not OK. By default, downtimes are ignored, unless explicitly told to be treated as acknowledgements through with the business_rule_downtime_as_ack set.

Returns True if all source problem are acknowledged, otherwise False

Return type bool

change_check_command(*command_params*)

Parameters **command_params** (*dict*) – command parameters

Returns

change_event_handler(*command_params*)

Parameters **command_params** (*dict*) – command parameters

Returns

change_snapshot_command(*command_params*)

Parameters **command_params** (*dict*) – command parameters

Returns

check_and_set_unreachability(*hosts*, *services*)
Check if all dependencies are down, if yes set this object as unreachable.

todo: this function do not care about execution_failure_criteria!

Parameters

- **hosts** (alignak.objects.host.Hosts) – hosts objects, used to get object in act_depend_of
- **services** (alignak.objects.service.Services) – services objects, used to get object in act_depend_of

Returns None

check_for_expire_acknowledge()
If have acknowledge and is expired, delete it

Returns None

check_for_flexible_downtime(*timeperiods*, *hosts*, *services*)
Enter in a downtime if necessary and raise start notification When a non Ok state occurs we try to raise a flexible downtime.

Parameters

- **timeperiods** (alignak.objects.timeperiod.Timeperiods) – Timeperiods objects, used for downtime period
- **hosts** (alignak.objects.host.Hosts) – hosts objects, used to enter downtime
- **services** (alignak.objects.service.Services) – services objects, used to enter downtime

Returns None

compensate_system_time_change(*difference*)
If a system time change occurs we have to update properties time related to reflect change

Parameters **difference** – difference between new time and old time

Returns None

consume_result(*chk*, *notification_period*, *hosts*, *services*, *timeperiods*, *macromodulations*, *checkmodulations*, *bi_modulations*, *res_modulations*, *checks*, *raise_log*)
Consume a check return and send action in return main function of reaction of checks like raise notifications

Special cases:

```
* is_flapping: immediate notif when problem
* is_in_scheduled_downtime: no notification
* is_volatile: notif immediately (service only)
```

Basically go through all cases (combination of last_state, current_state, attempt number) and do necessary actions (add attempt, raise notification., change state type.)

Parameters

- **chk** (*alignak.objects.check.Check*) – check to handle
- **notification_period** (alignak.objects.timeperiod.Timeperiod) – notification period for this host/service
- **hosts** (alignak.objects.host.Hosts) – hosts objects, used for almost every operation
- **services** (alignak.objects.service.Services) – services objects, used for almost every operation
- **timeperiods** (alignak.objects.timeperiod.Timeperiods) – Timeperiods objects, used for all kind of timeperiod (notif, check)
- **macromodulations** (*alignak.objects.macromodulation.Macromodulations*) – Macro modulations objects, used in commands (notif, check)
- **checkmodulations** (*alignak.objects.checkmodulation.Checkmodulations*) – Checkmodulations objects, used to change check command if necessary

- **bi_modulations** (*alignak.object.businessimpactmodulation.Businessimpactmodulations*) – business impact modulation are used when setting myself as problem
- **res_modulations** (*alignak.object.resultmodulation.Resultmodulations*) – result modulation are used to change the ouput of a check
- **checks** (*dict*) – checks dict, used to get checks_in_progress for the object

Returns Dependent checks

:rtype list[alignak.check.Check]

create_business_rules(*hosts*, *services*, *hostgroups*, *servicegroups*, *macromodulations*, *timeperiods*, *running=False*)

Create business rules if necessary (cmd contains bp_rule)

Parameters

- **hosts** (alignak.objects.host.Hosts) – Hosts object to look for objects
- **services** (alignak.objects.service.Services) – Services object to look for objects
- **running** (*bool*) – flag used in eval_cor_pattern function

Returns None

create_notifications(*n_type*, *notification_period*, *hosts*, *services*, *t_wished=None*, *author_data=None*)

Create a "master" notification here, which will later (immediately before the reactionner gets it) be split up in many "child" notifications, one for each contact.

Parameters

- **n_type** (*str*) – notification type ("PROBLEM", "RECOVERY" ...)
- **notification_period** (alignak.objects.timeperiod.Timeperiod) – notification period for this host/service
- **hosts** (alignak.objects.host.Hosts) – hosts objects, used to check if a notif is blocked
- **services** (alignak.objects.service.Services) – services objects, used to check if a notif is blocked
- **t_wished** (*int*) – time we want to notify
- **author_data** (*dict (containing author, author_name ad a comment)*) – notification author data (eg. for a downtime notification)

Returns None

current_event_id = 0

current_problem_id = 0

disable_active_checks(*checks*)

Disable active checks for this host/service Update check in progress with current object information

Parameters **checks** (*alignak.objects.check.Checks*) – Checks object, to change all checks in progress

Returns None

do_check_freshness(*hosts*, *services*, *timeperiods*, *macromodulations*, *checkmodulations*, *checks*, *when*)
Check freshness and schedule a check now if necessary.

This function is called by the scheduler if Alignak is configured to check the freshness.

It is called for hosts that have the freshness check enabled if they are only passively checked.

It is called for services that have the freshness check enabled if they are only passively checked and if their depending host is not in a freshness expired state (freshness_expiry = True).

A log is raised when the freshess expiry is detected and the item is set as freshness_expiry.

Parameters

- **hosts** (alignak.objects.host.Hosts) – hosts objects, used to launch checks
- **services** (alignak.objects.service.Services) – services objects, used launch checks
- **timeperiods** (alignak.objects.timeperiod.Timeperiods) – Timeperiods objects, used to get check_period
- **macromodulations** (*alignak.objects.macromodulation.Macromodulations*) – Macro modulations objects, used in commands (notif, check)
- **checkmodulations** (*alignak.objects.checkmodulation.Checkmodulations*) – Checkmodulations objects, used to change check command if necessary
- **checks** (*dict*) – checks dict, used to get checks_in_progress for the object

Returns A check or None

Return type None | object

do_i_raise_dependency(*status*, *inherit_parents*, *hosts*, *services*, *timeperiods*)
Check if this object or one of its dependency state (chk dependencies) match the status

Parameters

- **status** (*list*) – state list where dependency matters (notification failure criteria)
- **inherit_parents** (*bool*) – recurse over parents
- **hosts** (alignak.objects.host.Hosts) – hosts objects, used to raise dependency check
- **services** (alignak.objects.service.Services) – services objects, used to raise dependency check
- **timeperiods** (alignak.objects.timeperiod.Timeperiods) – Timeperiods objects, used for all kind of timeperiod (notif, check)

Returns True if one state matched the status list, otherwise False

Return type bool

downtimed
Simple property renaming for better API;)

fill_data_brok_from(*data*, *brok_type*)
Fill data brok dependent on the brok_type

Parameters

- **data** (*dict*) – data to fill
- **brok_type** – brok type

Type str

Returns None

get_business_rule_output(*hosts*, *services*, *macromodulations*, *timeperiods*)
Returns a status string for business rules based items formatted using business_rule_output_template attribute as template.

The template may embed output formatting for itself, and for its child (dependent) items. Child format string is expanded into the (and), using the string between brackets as format string.

Any business rule based item or child macro may be used. In addition, the $STATUS$, $SHORTSTATUS$ and $FULLNAME$ macro which name is common to hosts and services may be used to ease template writing.

Caution: only children in state not OK are displayed.

Example:

A business rule with a format string looking like "$STATUS$ [$($TATUS$: $HOSTNAME$,$SERVICEDESC$)$]"

Would return "CRITICAL [CRITICAL: host1,srv1 WARNING: host2,srv2]"

Parameters

- **hosts** (alignak.objects.host.Hosts) – Hosts object to look for objects
- **services** (alignak.objects.service.Services) – Services object to look for objects
- **macromodulations** (*alignak.objects.macromodulation.Macromodulations*) – Macromodulations object to look for objects
- **timeperiods** (alignak.objects.timeperiod.Timeperiods) – Timeperiods object to look for objects

Returns status for business rules

Return type str

get_data_for_checks(*hosts*)
Get data for a check

Returns list containing a single host (this one)

Return type list

get_data_for_event_handler(*hosts*)
Get data for an event handler

Returns list containing a single host (this one)

Return type list

get_data_for_notifications(*contact*, *notif*, *host_ref*)
Get data for a notification

Parameters

- **contact** – The contact to return

- **notif** – the notification to return

Returns list containing the host and the given parameters

Return type list

get_escalable_contacts(*notification*, *escalations*, *timeperiods*)
Get all contacts (uniq) from eligible escalations

Parameters

- **notification** (*alignak.objects.notification.Notification*) – Notification to get data from (notif number...)
- **escalations** (alignak.objects.escalation.Escalations) – Esclations objects, used to get escalation objects (contact, period)
- **timeperiods** (alignak.objects.timeperiod.Timeperiods) – Timeperiods objects, used to get escalation period

Returns Contact uuid list that can be notified for escalation

Return type list

get_event_handlers(*hosts*, *macromodulations*, *timeperiods*, *ext_cmd=False*)
Raise event handlers if NONE of the following conditions is met:

```
* externalcmd is False and event_handlers are disabled (globally or locally)
* externalcmd is False and object is in scheduled dowtime and no event␣
↪handlers in downtime
* self.event_handler and cls.global_event_handler are None
```

Parameters

- **hosts** (alignak.objects.host.Hosts) – hosts objects, used to get data for macros
- **macromodulations** (*alignak.objects.macromodulation.Macromodulations*) – Macro modulations objects, used in commands (notif, check)
- **timeperiods** (alignak.objects.timeperiod.Timeperiods) – Timeperiods objects, used for macros evaluation
- **ext_cmd** (*bool*) – tells if this function was called when handling an external_command.

Returns None

get_next_notification_time(*notif*, *escalations*, *timeperiods*)
Get the next notification time for a notification Take the standard notification_interval or ask for our escalation if one of them need a smaller value to escalade

Parameters

- **notif** (*alignak.objects.notification.Notification*) – Notification we need time
- **escalations** (alignak.objects.escalation.Escalations) – Esclations objects, used to get escalation objects (interval, period)
- **timeperiods** (alignak.objects.timeperiod.Timeperiods) – Timeperiods objects, used to get escalation period

Returns Timestamp of next notification

Return type int

get_perfdata_command(*hosts*, *macromodulations*, *timeperiods*)
Add event_handler to process performance data if necessary (not disabled)

Parameters **macromodulations** (*alignak.objects.macromodulation.Macromodulations*) – Macro modulations objects, used in commands (notif, check)

Returns None

get_snapshot(*hosts*, *macromodulations*, *timeperiods*)
Raise snapshot event handlers if NONE of the following conditions is met:

```
* snapshot_command is None
* snapshot_enabled is disabled
* snapshot_criteria does not matches current state
* last_snapshot > now - snapshot_interval * interval_length (previous
↪snapshot too early)
* snapshot_period is not valid
```

Parameters

- **hosts** (alignak.objects.host.Hosts) – hosts objects, used to get data for macros
- **macromodulations** (*alignak.objects.macromodulation.Macromodulations*) – Macro modulations objects, used in commands (notif, check)
- **timeperiods** (alignak.objects.timeperiod.Timeperiods) – Timeperiods objects, used for snapshot period and macros evaluation

Returns None

get_time_to_orphanage()
Get time to orphanage

```
* 0 : don't check for orphans
* non zero : number of secs that can pass before marking the check an orphan.
```

Returns integer with the meaning explained above

Return type int

is_blocking_notifications(*notification_period*, *hosts*, *services*, *n_type*, *t_wished*)
Check if a notification is blocked by item

Parameters

- **n_type** – notification type
- **t_wished** (*float*) – the time we should like to notify the host (mostly now)

Returns True if ONE of the above condition was met, otherwise False

Return type bool

is_correct()
Check if this object configuration is correct

```
* Check our own specific properties
* Call our parent class is_correct checker
```

Returns True if the configuration is correct, otherwise False

Return type bool

is_enable_action_dependent(*hosts*, *services*)
Check if dependencies states match dependencies statuses This basically means that a dependency is in a bad state and it can explain this object state.

Parameters

- **hosts** (alignak.objects.host.Hosts) – hosts objects, used to get object in act_depend_of
- **services** (alignak.objects.service.Services) – services objects, used to get object in act_depend_of

Returns True if all dependencies matches the status, false otherwise

Return type bool

is_escalable(*notification*, *escalations*, *timeperiods*)
Check if a notification can be escalated. Basically call is_eligible for each escalation

Parameters

- **notification** (*alignak.objects.notification.Notification*) – notification we would like to escalate
- **escalations** (alignak.objects.escalation.Escalations) – Esclations objects, used to get escalation objects (period)
- **timeperiods** (alignak.objects.timeperiod.Timeperiods) – Timeperiods objects, used to get escalation period

Returns True if notification can be escalated, otherwise False

Return type bool

is_max_attempts()
Check if max check attempt is reached

Returns True if self.attempt >= self.max_check_attempts, otherwise False

Return type bool

is_no_check_dependent(*hosts*, *services*, *timeperiods*)
Check if there is some host/service that this object depend on has a state in the status list .

Parameters

- **hosts** (alignak.objects.host.Hosts) – hosts objects, used to raise dependency check
- **services** (alignak.objects.service.Services) – services objects, used to raise dependency check
- **timeperiods** (alignak.objects.timeperiod.Timeperiods) – Timeperiods objects, used for all kind of timeperiod (notif, check)

Returns True if this object has a check dependency, otherwise False

Return type bool

is_state(*status*)
Return if status match the current item status

Parameters **status** (*str*) – status to compare. Usually comes from config files

Returns True

Return type bool

last_check
Simple property renaming for better API;)

launch_check(*timestamp*, *hosts*, *services*, *timeperiods*, *macromodulations*, *checkmodulations*, *checks*, *ref_check=None*, *force=False*, *dependent=False*)
Launch a check (command)

Parameters

- **timestamp** (*int*) –
- **checkmodulations** (*alignak.objects.checkmodulation.Checkmodulations*) – Checkmodulations objects, used to change check command if necessary
- **ref_check** –
- **force** (*bool*) –
- **dependent** (*bool*) –

Returns None or alignak.check.Check

Return type None | alignak.check.Check

macros = {'FULLNAME': 'get_full_name', 'SHORTSTATUS': ('get_short_status', ['hosts', '

manage_internal_check(*hosts*, *services*, *check*, *hostgroups*, *servicegroups*, *macromodulations*, *timeperiods*)
Manage internal commands such as

```
* bp_rule
* _internal_host_up
* _echo
```

Parameters

- **hosts** (alignak.objects.host.Hosts) – Used to create business rules
- **services** (alignak.objects.service.Services) – Used to create business rules
- **check** (*alignak.objects.check.Check*) – internal check to manage

Returns None

manage_stalking(*check*)
Check if the item need stalking or not (immediate recheck)

Parameters **check** (alignak.check.Check) – finished check (check.status == 'wait-consume')

Returns None

monitored
Simple property renaming for better API;)

next_check
Simple property renaming for better API;)

no_more_a_problem(*hosts*, *services*, *timeperiods*, *bi_modulations*)
Remove this objects as an impact for other schedulingitem.

Parameters

- **hosts** (alignak.objects.host.Hosts) – hosts objects, used to get impacts
- **services** (alignak.objects.service.Services) – services objects, used to get impacts
- **timeperiods** (alignak.objects.timeperiod.Timeperiods) – Timeperiods objects, used for update_business_impact_value
- **bi_modulations** (*alignak.object.businessimpactmodulation.Businessimpactmodulations*) – business impact modulation are used when setting myself as problem

Returns None

TODO: SchedulingItem object should not handle other schedulingitem obj. We should call obj.register* on both obj. This is 'Java' style

notification_is_blocked_by_contact(*notifways*, *timeperiods*, *notif*, *contact*)
Check if the notification is blocked by this contact.

Parameters

- **notif** (alignak.objects.contact.Contact) – notification created earlier
- **contact** – contact we want to notify

Returns True if the notification is blocked, False otherwise

Return type bool

old_properties = {'criticity': 'business_impact', 'normal_check_interval': 'check_in

prepare_notification_for_sending(*notif*, *contact*, *macromodulations*, *timeperiods*, *host_ref*)
Used by scheduler when a notification is ok to be sent (to reactionner). Here we update the command with status of now, and we add the contact to set of contact we notified. And we raise the log entry

Parameters

- **notif** (*alignak.objects.notification.Notification*) – notification to send
- **macromodulations** (*alignak.objects.macromodulation.Macromodulations*) – Macro modulations objects, used in the notification command
- **timeperiods** (alignak.objects.timeperiod.Timeperiods) – Timeperiods objects, used to get modulation period
- **host_ref** (*alignak.object.host.Host*) – reference host (used for a service)

Returns None

properties = {'action_url': <Property <class 'alignak.property.StringProp'>, default:

raise_acknowledge_log_entry()
Raise ACKNOWLEDGE STARTED entry Function defined in inherited objects (Host and Service)

Returns None

raise_alert_log_entry()
Raise ALERT entry Function defined in inherited objects (Host and Service)

Returns None

raise_check_result()
Raise ACTIVE CHECK RESULT entry Function defined in inherited objects (Host and Service)

Returns None

raise_dependencies_check(*ref_check*, *hosts*, *services*, *timeperiods*, *macromodulations*, *checkmodulations*, *checks*)
Get checks that we depend on if EVERY following conditions is met:

```
* timeperiod is valid
* dep.last_state_update < now - cls.cached_check_horizon (check of␣
↪dependency is "old")
```

Parameters

- **ref_check** (alignak.check.Check) – Check we want to get dependency from
- **hosts** (alignak.objects.host.Hosts) – hosts objects, used for almost every operation
- **services** (alignak.objects.service.Services) – services objects, used for almost every operation
- **timeperiods** (alignak.objects.timeperiod.Timeperiods) – Timeperiods objects, used for all kind of timeperiod (notif, check)
- **macromodulations** (*alignak.objects.macromodulation.Macromodulations*) – Macro modulations objects, used in commands (notif, check)
- **checkmodulations** (*alignak.objects.checkmodulation.Checkmodulations*) – Checkmodulations objects, used to change check command if necessary
- **checks** (*dict*) – checks dict, used to get checks_in_progress for the object

Returns check created and check in_checking

Return type dict

raise_event_handler_log_entry(*command*)
Raise EVENT HANDLER entry (critical level)

Parameters **command** (alignak.objects.command.Command) – Handler launched

Returns None

raise_flapping_start_log_entry(*change_ratio*, *threshold*)
Raise FLAPPING ALERT START entry (critical level)

Parameters

- **change_ratio** (*float*) – percent of changing state
- **threshold** (*float*) – threshold (percent) to trigger this log entry

Returns None

raise_flapping_stop_log_entry(*change_ratio*, *threshold*)
Raise FLAPPING ALERT STOPPED entry (critical level)

Parameters

- **change_ratio** (*float*) – percent of changing state
- **threshold** (*float*) – threshold (percent) to trigger this log entry

Returns None

raise_freshness_log_entry(*t_stale_by*)
Raise freshness alert entry (warning level)

Example ["The freshness period of host 'host_name' is expired] by 0d 0h 17m 6s (threshold=0d 1h 0m 0s). Attempt: 1 / 1. I'm forcing the state to freshness state (d / HARD)"

Parameters **t_stale_by** (*int*) – time in seconds the host has been in a stale state

Returns None

raise_notification_log_entry(*notif*, *contact*, *host_ref*)
Raise NOTIFICATION entry (critical level) :param notif: notification object created by service alert :type notif: alignak.objects.notification.Notification :return: None

raise_snapshot_log_entry(*command*)
Raise item SNAPSHOT entry (critical level) Format is : "ITEM SNAPSHOT: *self.get_name()*;*state*;*state_type*;*attempt*;

command.get_name()"

Example : "HOST SNAPSHOT: server;UP;HARD;1;notify-by-rss"

Parameters **command** (alignak.objects.command.Command) – Snapshot command launched

Returns None

raise_unacknowledge_log_entry()
Raise ACKNOWLEDGE STOPPED entry Function defined in inherited objects (Host and Service)

Returns None

register_a_problem(*prob*, *hosts*, *services*, *timeperiods*, *bi_modulations*)
Call recursively by potentials impacts so they update their source_problems list. But do not go below if the problem is not a real one for me like If I've got multiple parents for examples

Parameters

- **prob** (alignak.objects.schedulingitem.SchedulingItem) – problem to register
- **hosts** (alignak.objects.host.Hosts) – hosts objects, used to get object in act_depend_of_me
- **services** (alignak.objects.service.Services) – services objects, used to get object in act_depend_of_me

- **timeperiods** (alignak.objects.timeperiod.Timeperiods) – Timeperiods objects, used for all kind of timeperiod (notif, check)
- **bi_modulations** (*alignak.object.businessimpactmodulation.Businessimpactmodulations*) – business impact modulation are used when setting myself as problem

Returns list of host/service that are impacts

Return type list[*alignak.objects.schedulingitem.SchedulingItem*]

TODO: SchedulingItem object should not handle other schedulingitem obj. We should call obj.register* on both obj. This is 'Java' style

remove_in_progress_check(*check*)
Remove check from check in progress

Parameters **check** (*alignak.objects.check.Check*) – Check to remove

Returns None

remove_in_progress_notification(*notification*)
Remove a notification and mark them as zombie

Parameters **notification** (alignak.notification.Notification) – the notification to remove

Returns None

remove_in_progress_notifications(*master=True*)
Remove all notifications from notifications_in_progress

Preserves some specific notifications (downtime, ...)

Parameters

- **master** (*bool*) – remove master notifications only if True (default value)
- **force** (*bool*) – force remove all notifications except if False

:return:None

running_properties = {'acknowledgement': <Property <class 'alignak.property.StringPro

scatter_notification(*notif*, *contacts*, *notifways*, *timeperiods*, *macromodulations*, *escalations*, *host_ref*)
In create_notifications we created a notification master (eg. a template). When it's time to hand it over to the reactionner, this master notification needs to be split in several child notifications, one for each contact

To be more exact, one for each contact who is willing to accept notifications of this type and at this time

Parameters

- **notif** (*alignak.objects.notification.Notification*) – Notification to scatter
- **contacts** (alignak.objects.contact.Contacts) – Contacts objects, used to retreive contact for this object
- **notifways** (*alignak.object.notificationway.Notificationways*) – Notificationway objects, used to get notific commands
- **timeperiods** (alignak.objects.timeperiod.Timeperiods) – Timeperiods objects, used to check if notif are allowed at this time

- **macromodulations** (*alignak.objects.macromodulation.Macromodulations*) – Macro modulations objects, used in the notification command
- **escalations** (alignak.objects.escalation.Escalations) – Esclations objects, used to get escalated contacts
- **host_ref** (*alignak.object.host.Host*) – reference host (used for a service)

Returns child notifications

Return type list[alignak.objects.notification.Notification]

schedule(*hosts*, *services*, *timeperiods*, *macromodulations*, *checkmodulations*, *checks*, *force=False*, *force_time=None*)

Main scheduling function If a check is in progress, or active check are disabled, do not schedule a check. The check interval change with HARD state:

```
* SOFT: retry_interval
* HARD: check_interval
```

The first scheduling is evenly distributed, so all checks are not launched at the same time.

Parameters

- **hosts** (alignak.objects.host.Hosts) – hosts objects, used for almost every operation
- **services** (alignak.objects.service.Services) – services objects, used for almost every operation
- **timeperiods** (alignak.objects.timeperiod.Timeperiods) – Timeperiods objects, used for all kind of timeperiod (notif, check)
- **macromodulations** (*alignak.objects.macromodulation.Macromodulations*) – Macro modulations objects, used in commands (notif, check)
- **checkmodulations** (*alignak.objects.checkmodulation.Checkmodulations*) – Checkmodulations objects, used to change check command if necessary
- **checks** (*dict*) – checks dict, used to get checks_in_progress for the object
- **force** (*bool*) – tell if we forced this object to schedule a check
- **force_time** (*None | int*) – time we would like the check to be scheduled

Returns None

serialize()

This function serialize into a simple dict object. It is used when transferring data to other daemons over the network (http)

Here is the generic function that simply export attributes declared in the properties dictionary and the running_properties of the object.

Returns Dictionary containing key and value from properties and running_properties

Return type dict

set_impact_state()

We just go an impact, so we go unreachable But only if we enable this state change in the conf

Returns None

set_myself_as_problem(*hosts*, *services*, *timeperiods*, *bi_modulations*)
Raise all impact from my error. I'm setting myself as a problem, and I register myself as this in all hosts/services that depend_on_me. So they are now my impacts

Parameters

- **hosts** (alignak.objects.host.Hosts) – hosts objects, used to get impacts
- **services** (alignak.objects.service.Services) – services objects, used to get impacts
- **timeperiods** (alignak.objects.timeperiod.Timeperiods) – Timeperiods objects, used to get act_depend_of_me timeperiod
- **bi_modulations** (*alignak.object.businessimpactmodulation.Businessimpactmodulations*) – business impact modulations objects

Returns None

set_state_from_exit_status(*status*, *notif_period*, *hosts*, *services*)
Set the state with the status of a check. Also update last_state

Parameters

- **status** (*int*) – integer between 0 and 3
- **hosts** (alignak.objects.host.Hosts) – hosts objects, used for almost every operation
- **services** (alignak.objects.service.Services) – services objects, used for almost every operation

Returns None

set_unreachable()
Set unreachable: all our parents (dependencies) are not ok Unreachable is different from down/critical

:return:None

special_properties = []

unacknowledge_problem()
Remove the acknowledge, reset the flag. The comment is deleted

Returns None

unacknowledge_problem_if_not_sticky()
Remove the acknowledge if it is not sticky

Returns None

unregister_a_problem(*prob*)
Remove the problem from our problems list and check if we are still 'impacted'

Parameters **prob** (alignak.objects.schedulingitem.SchedulingItem) – problem to remove

Returns None

unset_impact_state()
Unset impact, only if impact state change is set in configuration

Returns None

update_business_impact_value(*hosts*, *services*, *timeperiods*, *bi_modulations*)
We update our 'business_impact' value with the max of the impacts business_impact if we got impacts. And save our 'configuration' business_impact if we do not have do it before If we do not have impacts, we revert our value

Parameters

- **hosts** (alignak.objects.host.Hosts) – hosts objects, used to get impacts
- **services** (alignak.objects.service.Services) – services objects, used to get impacts
- **timeperiods** (alignak.objects.timeperiod.Timeperiods) – Timeperiods objects, used to get modulation_period
- **bi_modulations** (*alignak.object.businessimpactmodulation.Businessimpactmodulations*) – business impact modulations objects

Returns None

TODO: SchedulingItem object should not handle other schedulingitem obj. We should call obj.register* on both obj. This is 'Java' style

update_event_and_problem_id()
Update current_event_id and current_problem_id Those attributes are used for macros (SERVICEPROBLEMID . . .)

Returns None

update_flapping(*notif_period*, *hosts*, *services*)
Compute the sample list (self.flapping_changes) and determine whether the host/service is flapping or not

Parameters

- **notif_period** (*alignak.object.timeperiod.Timeperiod*) – notification period object for this host/service
- **hosts** (alignak.objects.host.Hosts) – Hosts objects, used to create notification if necessary
- **services** (alignak.objects.service.Services) – Services objects, used to create notification if necessary

Returns None

Return type Nonetype

update_hard_unknown_phase_state()
Update in_hard_unknown_reach_phase attribute and was_in_hard_unknown_reach_phase UNKNOWN during a HARD state are not so important, and they should

not raise notif about it

Returns None

update_in_checking()
Update in_checking attribute. Object is in checking if we have checks in check_in_progress list

Returns None

update_notification_command(*notif*, *contact*, *macromodulations*, *timeperiods*, *host_ref=None*)

Update the notification command by resolving Macros And because we are just launching the notification, we can say that this contact has been notified

Parameters

- **notif** (*alignak.objects.notification.Notification*) – notification to send
- **contact** (*alignak.object.contact.Contact*) – contact for this host/service
- **macromodulations** (*alignak.objects.macromodulation.Macromodulations*) – Macro modulations objects, used in the notification command
- **timeperiods** (alignak.objects.timeperiod.Timeperiods) – Timeperiods objects, used to get modulation period
- **host_ref** (*alignak.object.host.Host*) – reference host (used for a service)

Returns None

class alignak.objects.schedulingitem.**SchedulingItems**(*items*, *index_items=True*, *parsing=True*)

Bases: *alignak.objects.commandcallitem.CommandCallItems*

Class to handle schedulingitems. It's mainly for configuration

add_act_dependency(*son_id*, *parent_id*, *notif_failure_criteria*, *dep_period*, *inherits_parents*)

Add a logical dependency for actions between two hosts or services.

Parameters

- **son_id** (*str*) – uuid of son host
- **parent_id** (*str*) – uuid of parent host
- **notif_failure_criteria** – notification failure criteria,

notification for a dependent host may vary :type notif_failure_criteria: list :param dep_period: dependency period. Timeperiod for dependency may vary :type dep_period: str | None :param inherits_parents: if this dep will inherit from parents (timeperiod, status) :type inherits_parents: bool :return:

add_chk_dependency(*son_id*, *parent_id*, *notif_failure_criteria*, *dep_period*, *inherits_parents*)

Add a logical dependency for checks between two hosts or services.

Parameters

- **son_id** (*str*) – uuid of son host/service
- **parent_id** (*str*) – uuid of parent host/service
- **notif_failure_criteria** – notification failure criteria,

notification for a dependent host may vary :type notif_failure_criteria: list :param dep_period: dependency period. Timeperiod for dependency may vary :type dep_period: str :param inherits_parents: if this dep will inherit from parents (timeperiod, status) :type inherits_parents: bool :return:

create_business_rules(*hosts*, *services*, *hostgroups*, *servicegroups*, *macromodulations*, *timeperiods*)

Loop on hosts or services and call SchedulingItem.create_business_rules

Parameters

- **hosts** (alignak.objects.host.Hosts) – hosts to link to
- **services** (alignak.objects.service.Services) – services to link to
- **hostgroups** (alignak.objects.hostgroup.Hostgroups) – hostgroups to link to
- **servicegroups** (alignak.objects.servicegroup.Servicegroups) – servicegroups to link to
- **macromodulations** (*alignak.objects.macromodulation.Macromodulations*) – macromodulations to link to
- **timeperiods** (alignak.objects.timeperiod.Timeperiods) – timeperiods to link to

Returns None

del_act_dependency(*son_id*, *parent_id*)
Remove act_dependency between two hosts or services.

TODO: do we really intend to remove dynamically ?

Parameters

- **son_id** (*str*) – uuid of son host/service
- **parent_id** (*str*) – uuid of parent host/service

Returns None

find_by_filter(*filters*, *all_items*)
Find items by filters

Parameters

- **filters** (*list*) – list of filters
- **all_items** – monitoring items

Type dict

Returns list of items

Return type list

alignak.objects.service module

This Class is the service one, s it manage all service specific thing. If you look at the scheduling part, look at the scheduling item class

class alignak.objects.service.**Service**(*params=None*, *parsing=True*)
Bases: *alignak.objects.schedulingitem.SchedulingItem*

Service class implements monitoring concepts for service. For example it defines parents, check_interval, check_command etc.

acknowledgement

acknowledgement_type

act_depend_of

act_depend_of_me

action_url

actions

active_checks_enabled

aggregation

alias

attempt

broks

business_impact

business_impact_modulations

business_rule

business_rule_downtime_as_ack

business_rule_host_notification_options

business_rule_output_template

business_rule_service_notification_options

business_rule_smart_notifications

check_command

check_flapping_recovery_notification

check_freshness

check_interval

check_period

checkmodulations

checks_in_progress

child_dependencies

chk_depend_of

chk_depend_of_me

comments

conf_is_correct

configuration_errors

configuration_warnings

contact_groups

contacts

current_event_id

int(x=0) -> int or long int(x, base=10) -> int or long

Convert a number or string to an integer, or return 0 if no arguments are given. If x is floating point, the conversion truncates towards zero. If x is outside the integer range, the function returns a long instead.

If x is not a number or if base is given, then x must be a string or Unicode object representing an integer literal in the given base. The literal can be preceded by '+' or '-' and be surrounded by whitespace. The base defaults to 10. Valid bases are 0 and 2-36. Base 0 means to interpret the base from the string as an integer literal. >>> int('0b100', base=0) 4

current_notification_id

current_notification_number

current_problem_id
int(x=0) -> int or long int(x, base=10) -> int or long

Convert a number or string to an integer, or return 0 if no arguments are given. If x is floating point, the conversion truncates towards zero. If x is outside the integer range, the function returns a long instead.

If x is not a number or if base is given, then x must be a string or Unicode object representing an integer literal in the given base. The literal can be preceded by '+' or '-' and be surrounded by whitespace. The base defaults to 10. Valid bases are 0 and 2-36. Base 0 means to interpret the base from the string as an integer literal. >>> int('0b100', base=0) 4

custom_views

customs

default_value

definition_order

display_name
Display_name if defined, else service_description

Returns service description or service display_name

Return type str

downtimes

duplicate(*host*)
For a given host, look for all copy we must create for for_each property

Parameters **host** (alignak.objects.host.Host) – alignak host object

Returns list

Return type list

duplicate_foreach

duration_sec

early_timeout

end_time

escalations

event_handler

event_handler_enabled

execution_time

fill_predictive_missing_parameters()
define state with initial_state

Returns None

first_notification_delay

flap_detection_enabled

flap_detection_options

flapping_changes

flapping_comment_id

freshness_expired

freshness_log_raised

freshness_state

freshness_threshold

get_ack_author_name()
Get the author of the acknowledgement

Returns author

Return type str

get_ack_comment()
Get the comment of the acknowledgement

Returns comment

Return type str

get_check_command()
Wrapper to get the name of the check_command attribute

Returns check_command name

Return type str

TODO: Move to util or SchedulingItem class

get_data_for_checks(*hosts*)
Get data for a check

Returns list containing the service and the linked host

Return type list

get_data_for_event_handler(*hosts*)
Get data for an event handler

Returns list containing the service and the linked host

Return type list

get_data_for_notifications(*contact*, *notif*, *host_ref*)
Get data for a notification

Parameters

- **contact** – The contact to return
- **notif** – the notification to return

Returns list containing the service, the host and the given parameters

Return type list

get_downtime()
Accessor to scheduled_downtime_depth attribute

Returns scheduled downtime depth

Return type str

TODO: Move to util or SchedulingItem class

get_duration()
Get duration formatted Format is : "HHh MMm SSs" Example : "10h 20m 40s"

Returns Formatted duration

Return type str

TODO: Move to util or SchedulingItem class

get_duration_sec()
Get duration in seconds. (cast it before returning)

Returns duration in seconds

Return type int

TODO: Move to util or SchedulingItem class

get_full_name()
Get the full name for debugging (host_name/service_description)

Returns service full name

Return type str

get_groupnames(*sgs*)
Get servicegroups list

Returns comma separated list of servicegroups

Return type str

get_host_tags(*hosts*)
Wrapper to access tags attribute of host attribute

Returns service tags (host one)

Return type alignak.objects.tag.Tags

get_hostgroups(*hosts*)
Wrapper to access hostgroups attribute of host attribute

Returns service hostgroups (host one)

Return type *alignak.objects.hostgroup.Hostgroups*

get_name()
Accessor to service_description attribute or name if first not defined

Returns service name

Return type str

get_service_tags()
Accessor to tags attribute

Returns service tags

Return type alignak.objects.tag.Tags

get_servicegroups()
Accessor to servicegroups attribute

Returns servicegroup list object of host

Return type list

get_short_status(*hosts*, *services*)
Get the short status of this host

Returns "O", "W", "C", "U', or "n/a" based on service state_id or business_rule state

Return type str

get_snapshot_command()
Wrapper to get the name of the snapshot_command attribute

Returns snapshot_command name

Return type str

get_status(*hosts*, *services*)
Get the status of this host

Returns "OK", "WARNING", "CRITICAL", "UNKNOWN" or "n/a" based on service state_id or business_rule state

Return type str

got_business_rule

has_been_checked

high_flap_threshold

host

host_dependency_enabled

host_name

hostgroup_name

icon_image

icon_image_alt

icon_set

impacts

imported_from

in_checking

in_hard_unknown_reach_phase

in_maintenance

in_scheduled_downtime

in_scheduled_downtime_during_last_check

initial_state

is_blocking_notifications(*notification_period*, *hosts*, *services*, *n_type*, *t_wished*)
Check if a notification is blocked by the service. Conditions are ONE of the following:

```
* enable_notification is False (global)
* not in a notification_period
* notifications_enable is False (local)
* notification_options is 'n' or matches the state ('UNKNOWN' <=> 'u' ...)
```

(include flapping and downtimes)

- state goes ok and type is 'ACKNOWLEDGEMENT' (no sense)
- scheduled_downtime_depth > 0 and flapping (host is in downtime)
- scheduled_downtime_depth > 1 and not downtime end (deep downtime)
- scheduled_downtime_depth > 0 and problem or recovery (host is in downtime)
- SOFT state of a problem (we raise notification ony on HARD state)
- ACK notification when already ACK (don't raise again ACK)
- not flapping notification in a flapping state
- business rule smart notifications is enabled and all its children have been acknowledged or are under downtime
- linked host is not up
- linked host is in downtime

Parameters

- **n_type** – notification type
- **t_wished** (*float*) – the time we should like to notify the host (mostly now)

Returns True if ONE of the above condition was met, otherwise False

Return type bool

TODO: Refactor this, a lot of code duplication with Host.is_blocking_notifications

is_correct()

Check if this object configuration is correct

```
* Check our own specific properties
* Call our parent class is_correct checker
```

Returns True if the configuration is correct, otherwise False

Return type bool

is_flapping

is_impact

is_problem

is_state(*status*)

Return True if status match the current service status

Parameters **status** (*str*) – status to compare ("o", "c", "w", "u", "x"). Usually comes from config files

Returns True if status <=> self.status, otherwise False

Return type bool

is_volatile

labels

last_check_command

last_chk

last_event_id

last_hard_state

last_hard_state_change

last_hard_state_id

last_notification

last_perf_data

last_problem_id

last_snapshot

last_state

last_state_change

last_state_id

last_state_type

last_state_update

last_time_critical

last_time_non_ok_or_up()
Get the last time the service was in a non-OK state

Returns the nearest last time the service was not ok

Return type int

last_time_ok

last_time_unknown

last_time_unreachable

last_time_warning

latency

long_output

low_flap_threshold

macromodulations

macros = {'FULLNAME': 'get_full_name', 'LASTSERVICECHECK': 'last_chk', 'LASTSERVICECRI

maintenance_period

manage_stalking(*check*)
Check if the service need stalking or not (immediate recheck) If one stalking_options matches the exit_status ('o' <=> 0 . . .) then stalk is needed Raise a log entry (info level) if stalk is needed

Parameters **check** (alignak.check.Check) – finished check (check.status == 'wait-consume')

Returns None

max_check_attempts

merge_host_contacts

modified_attributes

my_own_business_impact

my_type = 'service'

name

next_chk

notes

notes_url

notification_interval

notification_is_blocked_by_contact (*notifways*, *timeperiods*, *notif*, *contact*)
Check if the notification is blocked by this contact.

Parameters

- **notifways** (alignak.objects.notificationway.NotificationWays) – concerned notification ways
- **timeperiods** (alignak.objects.timeperiod.Timeperiods) – concerned timeperiods
- **notif** (alignak.notification.Notification) – notification created earlier
- **contact** (alignak.objects.contact.Contact) – contact we want to notify

Returns True if the notification is blocked, False otherwise

Return type bool

notification_options

notification_period

notifications_enabled

notifications_in_progress

notified_contacts

notified_contacts_ids

ok_up = u'OK'

old_properties = {'criticity': 'business_impact', 'hostgroup': 'hostgroup_name', 'ho

output

overall_state_id
Get the service overall state.

The service overall state identifier is the service status including: - the monitored state - the acknowledged state - the downtime state

The overall state is (prioritized): - a service is not monitored (5) - a service critical or unreachable (4) - a service warning or unknown (3) - a service downtimed (2) - a service acknowledged (1) - a service ok (0)

Note that services in unknown state are considered as warning, and unreachable ones are considered as critical!

Also note that the service state is considered only for HARD state type!

parallelize_check

parent_dependencies

passive_checks_enabled

pending_flex_downtime

percent_state_change

perf_data

poller_tag

problem_has_been_acknowledged

process_perf_data

processed_business_rule

properties = {'action_url': <Property <class 'alignak.property.StringProp'>, default:

raise_acknowledge_log_entry()
Raise SERVICE ACKNOWLEDGE STARTED entry (critical level)

Returns None

raise_alert_log_entry()
Raise SERVICE ALERT entry Format is : "SERVICE ALERT: *host.get_name()*;*get_name()*;*state*;*state_type*;*attempt*

;*output*"

Example : "SERVICE ALERT: server;Load;DOWN;HARD;1;I don't know what to say..."

Returns None

raise_cancel_downtime_log_entry()
Raise SERVICE DOWNTIME ALERT entry (critical level) Format is : "SERVICE DOWNTIME ALERT: *host_name*;*get_name()*;CANCELLED;

Service has entered a period of scheduled downtime"

Example ["SERVICE DOWNTIME ALERT: test_host_0;Load;CANCELLED;] Service has entered a period of scheduled downtime"

Returns None

raise_check_result()
Raise ACTIVE CHECK RESULT entry Example : "ACTIVE SERVICE CHECK: server;DOWN;HARD;1;I don't know what to say..."

Returns None

raise_enter_downtime_log_entry()
Raise SERVICE DOWNTIME ALERT entry (critical level) Format is : "SERVICE DOWNTIME ALERT: *host_name*;*get_name()*;STARTED;

Service has entered a period of scheduled downtime"

Example ["SERVICE DOWNTIME ALERT: test_host_0;Load;STARTED;] Service has entered a period of scheduled downtime"

Returns None

raise_event_handler_log_entry(*command*)
Raise SERVICE EVENT HANDLER entry (critical level) Format is : "SERVICE EVENT HANDLER: *host_name*;*self.get_name()*;*state*;*state_type*

;*attempt*;*command.get_name()*"

Example : "SERVICE EVENT HANDLER: server;Load;UP;HARD;1;notify-by-rss"

Parameters **command** (alignak.objects.command.Command) – Handler launched

Returns None

raise_exit_downtime_log_entry()
Raise SERVICE DOWNTIME ALERT entry (critical level) Format is : "SERVICE DOWNTIME ALERT: *host_name*;*get_name()*;STOPPED;

Service has entered a period of scheduled downtime"

Example ["SERVICE DOWNTIME ALERT: test_host_0;Load;STOPPED;] Service has entered a period of scheduled downtime"

Returns None

raise_flapping_start_log_entry(*change_ratio*, *threshold*)
Raise SERVICE FLAPPING ALERT START entry (critical level) Format is : "SERVICE FLAPPING ALERT: *host_name*;*self.get_name()*;STARTED;

Service appears to have started flapping (*change_ratio*% change >= *threshold*% threshold)"

Example ["SERVICE FLAPPING ALERT: server;Load;STARTED;] Service appears to have started flapping (50.6% change >= 50.0% threshold)"

Parameters

- **change_ratio** – percent of changing state
- **threshold** – threshold (percent) to trigger this log entry

Returns None

raise_flapping_stop_log_entry(*change_ratio*, *threshold*)
Raise SERVICE FLAPPING ALERT STOPPED entry (critical level) Format is : "SERVICE FLAPPING ALERT: *host_name*;*self.get_name()*;STOPPED;

Service appears to have started flapping (*change_ratio*% change >= *threshold*% threshold)"

Example ["SERVICE FLAPPING ALERT: server;Load;STOPPED;] Service appears to have started flapping (50.6% change >= 50.0% threshold)"

Parameters

- **change_ratio** (*float*) – percent of changing state
- **threshold** (*float*) – threshold (percent) to trigger this log entry

Returns None

raise_initial_state()
Raise SERVICE HOST ALERT entry (info level) Format is : "SERVICE HOST STATE: *host.get_name()*;*get_name()*;*state*;*state_type*

;*attempt*;*output*"

Example : "SERVICE HOST STATE: server;Load;DOWN;HARD;1;I don't know what to say..."

Returns None

raise_no_next_check_log_entry()
Raise no scheduled check entry (warning level) Format is : "I cannot schedule the check for the service '*get_name()*'

on host '*host_name*' because there is not future valid time"

Example ["I cannot schedule the check for the service 'Load' on host 'Server'] because there is not future valid time"

Returns None

raise_notification_log_entry(*notif*, *contact*, *host_ref*)
Raise SERVICE NOTIFICATION entry (critical level) Format is : "SERVICE NOTIFICATION: *contact.get_name()*;*host_name*;*self.get_name()*

;*state*;*command.get_name()*;*output*"

Example : "SERVICE NOTIFICATION: superadmin;server;Load;UP;notify-by-rss;no output"

Parameters **notif** (*alignak.objects.notification.Notification*) – notification object created by service alert

Returns None

raise_snapshot_log_entry(*command*)
Raise SERVICE SNAPSHOT entry (critical level) Format is : "SERVICE SNAPSHOT: *host_name*;*self.get_name()*;*state*;*state_type*;

attempt;*command.get_name()*"

Example : "SERVICE SNAPSHOT: server;Load;UP;HARD;1;notify-by-rss"

Parameters **command** (alignak.objects.command.Command) – Snapshot command launched

Returns None

raise_unacknowledge_log_entry()
Raise SERVICE ACKNOWLEDGE STOPPED entry (critical level)

Returns None

reactionner_tag

realm
Get the service realm... indeed it is the service's host one!

register

resultmodulations

retain_nonstatus_information

retain_status_information

retry_interval

return_code

running_properties = {'acknowledgement': <Property <class 'alignak.property.StringPro

s_time

scheduled_downtime_depth

service_dependencies

service_description

servicegroups

set_state_from_exit_status (*status*, *notif_period*, *hosts*, *services*)
Set the state in UP, WARNING, CRITICAL, UNKNOWN or UNREACHABLE according to the status of a check result.

Parameters **status** (*int*) – integer between 0 and 4

Returns None

should_be_scheduled

snapshot_command

snapshot_criteria

snapshot_enabled

snapshot_interval

snapshot_period

source_problems

special_properties = 'service_description'

stalking_options

start_time

state

state_before_hard_unknown_reach_phase

state_before_impact

state_changed_since_impact

state_id

state_id_before_impact

state_type

state_type_id

tags

time_to_orphanage

timeout

topology_change

trending_policies

u_time

unique_key
Unique key for this service

Returns Tuple with host_name and service_description

Return type tuple

use

was_in_hard_unknown_reach_phase

class alignak.objects.service.**Services**(*items*, *index_items=True*, *parsing=True*)
Bases: *alignak.objects.schedulingitem.SchedulingItems*

Class for the services lists. It's mainly for configuration

add_template(*tpl*)
Adds and index a template into the *templates* container.

This implementation takes into account that a service has two naming attribute: *host_name* and *service_description*.

Parameters **tpl** – The template to add

Returns None

apply_dependencies(*hosts*)
Wrapper to loop over services and call Service.fill_daddy_dependency()

Returns None

apply_implicit_inheritance(*hosts*)
Apply implicit inheritance for special properties: contact_groups, notification_interval , notification_period So service will take info from host if necessary

Parameters **hosts** (alignak.objects.host.Hosts) – hosts list needed to look for a simple host

Returns None

apply_inheritance()

For all items and templates inherit properties and custom variables.

Returns None

clean()
Remove services without host object linked to

Note that this should not happen!

Returns None

delete_services_by_id(*ids*)
Delete a list of services

Parameters **ids** (*list*) – ids list to delete

Returns None

explode(*hosts*, *hostgroups*, *contactgroups*, *servicegroups*, *servicedependencies*)
Explodes services, from host, hostgroups, contactgroups, servicegroups and dependencies.

Parameters

- **hosts** (*[alignak.object.host.Host]*) – The hosts container

- **hostgroups** (*[alignak.object.hostgroup.Hostgroup]*) – The hosts goups container
- **contactgroups** (*[alignak.object.contactgroup.Contactgroup]*) – The contacts goups container
- **servicegroups** (*[alignak.object.servicegroup.Servicegroup]*) – The services goups container
- **servicedependencies** (*[alignak.object.servicedependency.Servicedependency]*) – The services dependencies container

Returns None

explode_services_duplicates(*hosts*, *service*)
Explodes services holding a *duplicate_foreach* clause.

Parameters

- **hosts** (alignak.objects.host.Hosts) – The hosts container
- **service** (alignak.objects.service.Service) – The service to explode

explode_services_from_hosts(*hosts*, *service*, *hnames*)
Explodes a service based on a list of hosts.

Parameters

- **hosts** – The hosts container
- **service** – The base service to explode
- **hnames** (*str*) – The host_name list to explode service on

Returns None

explode_services_from_templates(*hosts*, *service_template*)
Explodes services from templates. All hosts holding the specified templates are bound with the service.

Parameters

- **hosts** (alignak.objects.host.Hosts) – The hosts container.
- **service_template** (alignak.objects.service.Service) – The service to explode.

Returns None

fill_predictive_missing_parameters()
Loop on services and call Service.fill_predictive_missing_parameters()

Returns None

find_srv_by_name_and_hostname(*host_name*, *sdescr*)
Get a specific service based on a host_name and service_description

Parameters

- **host_name** (*str*) – host name linked to needed service
- **sdescr** (*str*) – service name we need

Returns the service found or None

Return type *alignak.objects.service.Service*

find_srvs_by_hostname(*host_name*)
Get all services from a host based on a host_name

Parameters **host_name** (*str*) – the host name we want services

Returns list of services

Return type list[*alignak.objects.service.Service*]

inner_class
alias of *Service*

linkify(*hosts*, *commands*, *timeperiods*, *contacts*, *resultmodulations*, *businessimpactmodulations*, *escalations*, *servicegroups*, *checkmodulations*, *macromodulations*)
Create link between objects:

```
* service -> host
* service -> command
* service -> timeperiods
* service -> contacts
```

Parameters

- **hosts** (alignak.objects.host.Hosts) – hosts to link
- **timeperiods** (alignak.objects.timeperiod.Timeperiods) – timeperiods to link
- **commands** (alignak.objects.command.Commands) – commands to link
- **contacts** (alignak.objects.contact.Contacts) – contacts to link
- **resultmodulations** (alignak.objects.resultmodulation.Resultmodulations) – resultmodulations to link
- **businessimpactmodulations** (alignak.objects.businessimpactmodulation.Businessimpactmodulations) – businessimpactmodulations to link
- **escalations** (alignak.objects.escalation.Escalations) – escalations to link
- **servicegroups** (alignak.objects.servicegroup.Servicegroups) – servicegroups to link
- **checkmodulations** (*alignak.objects.checkmodulation.Checkmodulations*) – checkmodulations to link
- **macromodulations** (*alignak.objects.macromodulation.Macromodulations*) – macromodulations to link

Returns None

linkify_s_by_hst(*hosts*)
Link services with their parent host

Parameters **hosts** (alignak.objects.host.Hosts) – Hosts to look for simple host

Returns None

linkify_s_by_sg(*servicegroups*)
Link services with servicegroups

Parameters **servicegroups** (alignak.objects.servicegroup.Servicegroups) – Servicegroups

Returns None

name_property = 'unique_key'

optimize_service_search(*hosts*)
Setter for hosts attribute

Parameters **hosts** (alignak.objects.host.Hosts) – value to set

Returns

override_properties(*hosts*)
Handle service_overrides property for hosts ie : override properties for relevant services

Parameters **hosts** (alignak.objects.host.Hosts) – hosts we need to apply override properties

Returns None

static register_service_dependencies(*service*, *servicedependencies*)
Registers a service dependencies.

Parameters

- **service** – The service to register
- **servicedependencies** – The servicedependencies container

Returns None

static register_service_into_servicegroups(*service*, *servicegroups*)
Registers a service into the service groups declared in its *servicegroups* attribute.

Parameters

- **service** – The service to register
- **servicegroups** – The servicegroups container

Returns None

alignak.objects.servicedependency module

This module provides Servicedependency and Servicedependencies classes that implements dependencies between services. Basically used for parsing.

class alignak.objects.servicedependency.**Servicedependencies**(*items*, *index_items=True*, *parsing=True*)

Bases: *alignak.objects.item.Items*

Servicedependencies manage a list of Servicedependency objects, used for parsing configuration

add_service_dependency(*dep_host_name*, *dep_service_description*, *par_host_name*, *par_service_description*)
Instantiate and add a Servicedependency object to the items dict:

```
* notification criteria is "u,c,w"
* inherits_parent is True
```

Parameters

- **dep_host_name** (*str*) – dependent host name
- **dep_service_description** (*str*) – dependent service description

- **par_host_name** (*str*) – host name
- **par_service_description** (*str*) – service description

Returns None

delete_servicesdep_by_id(*ids*)
Delete a list of servicedependency

Parameters **ids** (*list*) – ids list to delete

Returns None

explode(*hostgroups*)
Explode all service dependency for each member of hostgroups Each member of dependent hostgroup or hostgroup in dependency have to get a copy of service dependencies (quite complex to parse)

Parameters **hostgroups** (alignak.objects.hostgroup.Hostgroups) – used to look for hostgroup

Returns None

explode_hostgroup(*svc_dep*, *hostgroups*)
Explode a service dependency for each member of hostgroup

Parameters

- **svc_dep** (alignak.objects.servicedependency.Servicedependency) – service dependency to explode
- **hostgroups** (alignak.objects.hostgroup.Hostgroups) – used to find hostgroup objects

:return:None

inner_class
alias of *Servicedependency*

is_correct()
Check if this servicedependency configuration is correct

```
* Check our own specific properties
* Call our parent class is_correct checker
```

Returns True if the configuration is correct, otherwise False

Return type bool

linkify(*hosts*, *services*, *timeperiods*)
Create link between objects:

```
* servicedependency -> host
* servicedependency -> service
* servicedependency -> timeperiods
```

Parameters

- **hosts** (alignak.objects.host.Hosts) – hosts to link
- **services** (alignak.objects.service.Services) – services to link
- **timeperiods** (alignak.objects.timeperiod.Timeperiods) – timeperiods to link

Returns None

linkify_s_by_sd(*services*)
Add dependency in service objects

Returns None

linkify_sd_by_s(*hosts*, *services*)
Replace dependent_service_description and service_description in service dependency by the real object

Parameters

- **hosts** (alignak.objects.host.Hosts) – host list, used to look for a specific one
- **services** (alignak.objects.service.Services) – service list to look for a specific one

Returns None

linkify_sd_by_tp(*timeperiods*)
Replace dependency_period by a real object in service dependency

Parameters **timeperiods** (alignak.objects.timeperiod.Timeperiods) – list of timeperiod, used to look for a specific one

Returns None

class alignak.objects.servicedependency.**Servicedependency**(*params=None*, *parsing=True*)

Bases: *alignak.objects.item.Item*

Servicedependency class is a simple implementation of service dependency as defined in a monitoring context (dependency period, notification_failure_criteria ..)

get_name()
Get name based on 4 class attributes Each attribute is substituted by '' if attribute does not exist

Returns dependent_host_name/dependent_service_description..host_name/service_description

Return type str

TODO: Clean this function (use format for string)

my_type = 'servicedependency'

properties = {'definition_order': <Property <class 'alignak.property.IntegerProp'>, d

alignak.objects.serviceescalation module

This module provides Serviceescalation and Serviceescalations classes that implements service escalation for notification. Basically used for parsing.

class alignak.objects.serviceescalation.**Serviceescalation**(*params=None*, *parsing=True*)

Bases: *alignak.objects.item.Item*

Serviceescalation class is used to implement notification escalation for services

TODO: Why this class does not inherit from alignak.objects.Escalation. Maybe we can merge it

my_type = 'serviceescalation'

properties = {'contact_groups': <Property <class 'alignak.property.ListProp'>, defaul

class alignak.objects.serviceescalation.**Serviceescalations**(*items*, *index_items=True*, *parsing=True*)

Bases: *alignak.objects.item.Items*

Serviceescalations manage a list of Serviceescalation objects, used for parsing configuration

explode(*escalations*)

Create instance of Escalation for each ServiceEscalation object

Parameters **escalations** (*alignak.objects.escalation.Escalations*) – list of escalation, used to add new ones

Returns None

inner_class

alias of *Serviceescalation*

name_property = ''

alignak.objects.serviceextinfo module

This is the main class for the Service ext info. In fact it's mainly about the configuration part. Parameters are merged in Service so it's no use in running part

class alignak.objects.serviceextinfo.**ServiceExtInfo**(*params=None*, *parsing=True*)

Bases: *alignak.objects.genericextinfo.GenericExtInfo*

ServiceExtInfo class is made to handle some parameters of SchedulingItem:

```
* notes
* notes_url
* icon_image
* icon_image_alt
```

TODO: Is this class really necessary?

definition_order

host_name

icon_image

icon_image_alt

imported_from

macros = {'SERVICEACTIONURL': 'action_url', 'SERVICEDESC': 'service_description', 'SER

my_type = 'serviceextinfo'

name

notes

notes_url

properties = {'definition_order': <Property <class 'alignak.property.IntegerProp'>, d

register

service_description

use

class alignak.objects.serviceextinfo.**ServicesExtInfo**(*items*, *index_items=True*, *parsing=True*)

Bases: *alignak.objects.item.Items*

ServicesExtInfo manage ServiceExtInfo and propagate properties (listed before) into Services if necessary

inner_class
alias of *ServiceExtInfo*

merge(*services*)
Merge extended host information into services

Parameters **services** (alignak.objects.service.Services) – services list, to look for a specific one

Returns None

static merge_extinfo(*service*, *extinfo*)
Merge extended host information into a service

Parameters

- **service** (alignak.objects.service.Service) – the service to edit
- **extinfo** (alignak.objects.serviceextinfo.ServiceExtInfo) – the external info we get data from

Returns None

name_property = 'host_name'

alignak.objects.servicegroup module

This module provide Servicegroup and Servicegroups classes used to group services

class alignak.objects.servicegroup.**Servicegroup**(*params=None*, *parsing=True*)

Bases: *alignak.objects.itemgroup.Itemgroup*

Class to manage a servicegroup A servicegroup is used to group services

get_name()
Get the group name

get_servicegroup_members()
Get the groups members of the group

Returns list of services

Return type list | str

get_services()
Get the services of the group

Returns list of services (members)

Return type list

get_services_by_explosion(*servicegroups*)
Get all services of this servicegroup and add it in members container

Parameters **servicegroups** (alignak.objects.servicegroup.Servicegroups) – servicegroups object

Returns return empty string or list of members

Return type str or list

group_members_property = 'servicegroup_members'

macros = {'SERVICEGROUPACTIONURL': 'action_url', 'SERVICEGROUPALIAS': 'alias', 'SERVIC

members_property = 'members'

my_type = 'servicegroup'

properties = {'action_url': <Property <class 'alignak.property.StringProp'>, default:

class alignak.objects.servicegroup.**Servicegroups**(*items*, *index_items=True*, *parsing=True*)

Bases: *alignak.objects.itemgroup.Itemgroups*

Class to manage all servicegroups

add_member(*service_name*, *servicegroup_name*)

Add a member (service) to this servicegroup

Parameters

- **service_name** (*str*) – member (service) name
- **servicegroup_name** (*str*) – servicegroup name

Returns None

explode()

Get services and put them in members container

Returns None

get_members_of_group(*gname*)

Get all members of a group which name is given in parameter

Parameters **gname** (*str*) – name of the group

Returns list of the services in the group

Return type list[*alignak.objects.service.Service*]

inner_class

alias of *Servicegroup*

linkify(*hosts*, *services*)

Link services with host

Parameters

- **hosts** (alignak.objects.host.Hosts) – hosts object
- **services** (alignak.objects.service.Services) – services object

Returns None

linkify_servicegroups_services(*hosts*, *services*)

We just search for each host the id of the host and replace the name by the id TODO: very slow for high services, so search with host list, not service one

Parameters

- **hosts** (alignak.objects.host.Hosts) – hosts object
- **services** (alignak.objects.service.Services) – services object

Returns None

```
name_property = 'servicegroup_name'
```

alignak.objects.timeperiod module

This module provide Timeperiod class used to define time periods to do action or not if we are in right period

class alignak.objects.timeperiod.**Timeperiod**(*params=None*, *parsing=True*)
Bases: *alignak.objects.item.Item*

Class to manage a timeperiod A timeperiod is defined with range time (hours) of week to do action and add day exceptions (like non working days)

apply_inheritance()
Inherite no properties and no custom variables for timeperiod

Returns None

check_and_log_activation_change()
Will look for active/un-active change of timeperiod. In case it change, we log it like: [1327392000] TIMEPERIOD TRANSITION: <name>;<from>;<to>

States of is_active: -1: default value when start 0: when timeperiod end 1: when timeperiod start

Returns None or a brok if TP changed

check_exclude_rec()
Check if this timeperiod is tagged

Returns if tagged return false, if not true

Return type bool

clean_cache()
Clean cache with entries older than now because not used in future ;)

Returns None

explode()
Try to resolve all unresolved elements

Returns None

fill_data_brok_from(*data*, *brok_type*)
Add timeperiods from brok

Parameters

- **data** (*dict*) – timeperiod dictionnary
- **brok_type** (*string*) – brok type

Returns None

find_next_invalid_time_from_cache(*timestamp*)
Get the next invalid time from cache

Parameters **timestamp** (*int*) – number of seconds

Returns Nothing or time in seconds

Return type None or int

find_next_valid_time_from_cache(*timestamp*)
Get the next valid time from cache

Parameters **timestamp** (*int*) – number of seconds

Returns Nothing or time in seconds

Return type None or int

get_min_from_t(*timestamp*)
Get the first time > timestamp which is valid

Parameters **timestamp** (*int*) – number of seconds

Returns number of seconds

Return type int

TODO: not used, so delete it

get_name()
Get the name of the timeperiod

Returns the timeperiod name string

Return type str

get_next_invalid_time_from_t(*timestamp*)
Get the next invalid time

Parameters **timestamp** (*int or float*) – timestamp in seconds (of course)

Returns timestamp of next invalid time

Return type int or float

get_next_valid_time_from_t(*timestamp*)
Get next valid time. If it's in cache, get it, otherwise define it. The limit to find it is 1 year.

Parameters **timestamp** (*int or float*) – number of seconds

Returns Nothing or time in seconds

Return type None or int

get_not_in_min_from_t(*first*)

Returns None

TODO: not used, so delete it

get_raw_import_values()
Get some properties of timeperiod (timeperiod is a bit different from classic item)

TODO: never called anywhere, still useful?

Returns a dictionnary of some properties

Return type dict

is_correct()
Check if this object configuration is correct

```
* Check if dateranges of timeperiod are valid
* Call our parent class is_correct checker
```

Returns True if the configuration is correct, otherwise False if at least one daterange

is not correct :rtype: bool

is_time_valid(*timestamp*)
Check if a time is valid or not

Returns time is valid or not

Return type bool

linkify(*timeperiods*)
Will make timeperiod in exclude with id of the timeperiods

Parameters **timeperiods** – Timeperiods object

Returns None

my_type = 'timeperiod'

properties = {'activated_once': <Property <class 'alignak.property.BoolProp'>, defaul

resolve_daterange(*dateranges*, *entry*)
Try to solve dateranges (special cases)

Parameters

- **dateranges** (*list*) – dateranges
- **entry** (*string*) – property of timeperiod

Returns None

running_properties = {'conf_is_correct': <Property <class 'alignak.property.BoolProp'

serialize()
This function serialize into a simple dict object. It is used when transferring data to other daemons over the network (http)

Here we directly return all attributes

Returns json representation of a Timeperiod

Return type dict

class alignak.objects.timeperiod.**Timeperiods**(*items*, *index_items=True*, *parsing=True*)
Bases: *alignak.objects.item.Items*

Class to manage all timeperiods A timeperiod is defined with range time (hours) of week to do action and add day exceptions (like non working days)

apply_inheritance()
The only interesting property to inherit is exclude

Returns None

explode()
Try to resolve each timeperiod

Returns None

get_unresolved_properties_by_inheritance(*timeperiod*)
Fill full properties with template if needed for the unresolved values (example: sunday ETCETC) :return: None

inner_class
alias of *Timeperiod*

is_correct()
check if each properties of timeperiods are valid

Returns True if is correct, otherwise False

Return type bool

linkify()
Check exclusion for each timeperiod

Returns None

name_property = 'timeperiod_name'

Module contents

The objects package contains the definition of the classes for the different objects that can be declared in configuration files.

2.2 Submodules

2.3 alignak.acknowledge module

This module provides Acknowledge class that implements acknowledgment for notification. Basically used for parsing.

class alignak.acknowledge.**Acknowledge**(*params=None*, *parsing=False*)
Bases: *alignak.alignakobject.AlignakObject*

Allows you to acknowledge the current problem for the specified service. By acknowledging the current problem, future notifications (for the same service state) are disabled.

If the acknowledge is "sticky", the acknowledgement will remain until the service returns to an OK state. Otherwise the acknowledgement will automatically be removed when the service state changes.

If the acknowledge is "notify", a notification will be sent out to contacts indicating that the current service problem has been acknowledged and when the acknowledge is cleared.

get_expire_brok(*host_name*, *service_name=''*)
Get an expire acknowledge brok

Returns brok with wanted data

Return type *alignak.brok.Brok*

get_raise_brok(*host_name*, *service_name=''*)
Get a start acknowledge brok

Parameters

- **host_name** –
- **service_name** –

Returns brok with wanted data

Return type *alignak.brok.Brok*

my_type = 'acknowledge'

properties = {'author': <Property <class 'alignak.property.StringProp'>, default: u'

serialize()
This function serialize into a simple dict object. It is used when transferring data to other daemons over the network (http)

Here we directly return all attributes

Returns json representation of a Acknowledge

Return type dict

2.4 alignak.action module

This module provides a system-independent Action class. Action class is used for handling check and notification execution (handle output, execute process, kill process..)

class alignak.action.**Action**(*params=None*, *parsing=False*)

Bases: alignak.action.ActionBase

Action class for *NIX systems

properties = {'_in_timeout': <Property <class 'alignak.property.BoolProp'>, default:

2.5 alignak.alignakobject module

This module contains only a common class for all object created in Alignak: AlignakObject.

class alignak.alignakobject.**AlignakObject**(*params=None*, *parsing=True*)

Bases: object

This class provides a generic way to instantiate alignak objects. Attributes are serialized dynamically, whether we un-serialize them create them at run / parsing time

fill_default()

Define the object properties with a default value when the property is not yet defined

Returns None

macros = {}

properties = {}

serialize()

This function serializes into a simple dictionary object.

It is used when transferring data to other daemons over the network (http)

Here is the generic function that simply export attributes declared in the properties dictionary of the object.

Note that a SetProp property will be serialized as a list.

Returns Dictionary containing key and value from properties

Return type dict

alignak.alignakobject.**get_a_new_object_id**()

Get a new Alignak object identifier. Uses the uuid version 1 generator

Rtype uuid bytes

Returns uuid

2.6 alignak.autoslots module

The AutoSlots Class is a MetaClass: it manages how other classes are created (Classes, not instances of theses classes). Here its role is to create the __slots__ list of the class with all properties of Class.properties and Class.running_properties so we do not have to add manually all properties to the __slots__ list when we add a new entry

class alignak.autoslots.**AutoSlots**
Bases: type

AutoSlots inherit from type, it's compulsory for metaclass statement

2.7 alignak.basemodule module

This python module contains the class BaseModule that alignak modules will subclass

class alignak.basemodule.**BaseModule**(*mod_conf*)
Bases: object

This is the base class for the Alignak modules. Modules can be used by the different Alignak daemons for different tasks. Example of task that an Alignak module can do: - load additional configuration objects. - recurrently save hosts/services status/perfdata information in different format. - ...

alias
Module name may be stored in an alias property Stay compatible with older modules interface

clear_queues(*manager*)
Release the resources associated to the queues of this instance

Parameters **manager** (*None | object*) – Manager() object

Returns None

create_queues(*manager=None*)
Create the shared queues that will be used by alignak daemon process and this module process. But clear queues if they were already set before recreating new one.

Note: If manager is None, then we are running the unit tests for the modules and we must create some queues for the external modules without a SyncManager

Parameters **manager** (*None | object*) – Manager() object

Returns None

do_loop_turn()
For external modules only: implement in this method the body of you main loop

Returns None

do_stop()
Called just before the module will exit Put in this method all you need to cleanly release all open resources used by your module

Returns None

get_name()
Wrapper to access name attribute

Returns module name

Return type str

init()
Handle this module "post" init ; just before it'll be started.

This function initializes the module instance. If False is returned, the modules manager will periodically retry an to initialize the module. If an exception is raised, the module will be definitely considered as dead :/

This function must be present and return True for Alignak to consider the module as loaded and fully functional.

Returns True / False according to initialization succeeded or not

Return type bool

kill()
Sometime terminate() is not enough, we must "help" external modules to die...

Returns None

main()
Main function of BaseModule

Returns None

manage_brok(*brok*)
Request the module to manage the given brok. There are a lot of different possible broks to manage. The list is defined in the Brok class.

An internal module may redefine this function or, easier, define only the function for the brok it is interested with. Hence a module interested in the *service_check_result* broks will only need to define a function named as *manage_service_check_result_brok*

Parameters **brok** –

Returns

Return type

manage_signal(*sig*, *frame*)
Generic function to handle signals

Only called when the module process received SIGINT or SIGKILL.

Set interrupted attribute to True, self.process to None and returns

Parameters

- **sig** – signal sent
- **frame** – frame before catching signal

Returns None

set_exit_handler(*sigs=None*)
Set the signal handler to manage_signal (defined in this class)

Only set handlers for: - signal.SIGTERM, signal.SIGINT - signal.SIGUSR1, signal.SIGUSR2 - signal.SIGHUP

Returns None

set_loaded_into(*daemon_name*)
Setter for loaded_into attribute Used to know what daemon has loaded this module

Parameters **daemon_name** (*str*) – value to set

Returns None

set_proctitle(*name*)
Wrapper for setproctitle method

Parameters **name** (*str*) – module alias

Returns None

set_signal_handler(*sigs=None*)
Set the signal handler to manage_signal (defined in this class)

Only set handlers for: - signal.SIGTERM, signal.SIGINT - signal.SIGUSR1, signal.SIGUSR2 - signal.SIGHUP

Returns None

start(*http_daemon=None*)
Actually restart the process if the module is external Try first to stop the process and create a new Process instance with target start_module. Finally start process.

Parameters **http_daemon** (*None | object*) – Not used here but can be used in other modules

Returns None

start_module()
Wrapper for _main function. Catch and raise any exception occurring in the main function

Returns None

stop_process()
Request the module process to stop and release it

Returns None

want_brok(*b*)
Generic function to check if the module need a specific brok In this case it is always True

Parameters **b** (alignak.brok.Brok) – brok to check

Returns True if the module wants the brok, False otherwise

Return type bool

work()
module "main" method. Only used by external modules.

Returns None

2.8 alignak.borg module

Borg module provides Borg class. Used only for MacroSolver

class alignak.borg.**Borg**
Bases: object

Borg class define a simple __shared_state class attribute. __dict__ points to this value when calling __init__

This is used to make a Singleton-like pattern with a python object that inherits from the Borg.

The Singleton design pattern (DP) has a catchy name, but the wrong focus – on identity rather than on state. The Borg design pattern has all instances share state instead, and Python makes it, literally, a snap.

2.9 alignak.brok module

Brok module provide Brok class which is basically event for Alignak. Brok are filled depending on their type (check_result, initial_state ...)

class alignak.brok.**Brok**(*params*, *parsing=True*)

Bases: object

A Brok is a piece of information exported by Alignak to the Broker. Broker can do whatever he wants with it.

A specific type of Brok exists when the type is monitoring_log. This Brok contains a monitoring event (alert, notification, ...) information

Broks types: - log - monitoring_log

- notification_raise
- acknowledge_raise
- downtime_raise
- acknowledge_expire
- downtime_expire
- initial_host_status, initial_service_status, initial_contact_status
- initial_broks_done
- update_host_status, update_service_status, initial_contact_status
- host_check_result, service_check_result
- host_next_schedule, service_next_scheduler
- host_snapshot, service_snapshot
- unknown_host_check_result, unknown_service_check_result
- program_status, initial program status
- update_program_status, program status updated (raised on each scheduler loop)
- clean_all_my_instance_id
- new_conf

get_event()

This function returns an Event from a Brok

If the type is monitoring_log then the Brok contains a monitoring event (alert, notification, ...) information. This function will return a tuple with the creation time, the level and message information

Returns tuple with date, level and message

Return type tuple

my_type = 'brok'

prepare()

Un-serialize data from data attribute and add instance_id key if necessary

Returns None

serialize()

This function serialize into a simple dict object. It is used when transferring data to other daemons over the network (http)

Here we directly return all attributes

Returns json representation of a Brok

Return type dict

2.10 alignak.check module

This module provides Check class which is a simple abstraction for monitoring checks

class alignak.check.**Check**(*params=None*, *parsing=False*)

Bases: *alignak.action.Action*

Check class implements monitoring concepts of checks :(status, state, output) Check instance are used to store monitoring plugins data (exit status, output) and used by schedule to raise alert, reschedule check etc.

get_return_from(*check*)

Update check data from action (notification for instance)

Parameters **check** (alignak.action.Action) – action to get data from

Returns None

is_dependent()

Getter for dependency_check attribute

Returns True if this check was created for a dependent one, False otherwise

Return type bool

my_type = 'check'

properties = {'_in_timeout': <Property <class 'alignak.property.BoolProp'>, default:

serialize()

This function serializes into a simple dict object.

The only usage is to send to poller, and it does not need to have the depend_on and depend_on_me properties.

Returns json representation of a Check

Return type dict

set_type_active()

Set this check as an active one (indeed, not passive)

Returns None

set_type_passive()

Set this check as a passive one

Returns None

2.11 alignak.commandcall module

This modules provide CommandCall class which is a abstraction for dealing with command line (resolve macro, parse commands etc)

class alignak.commandcall.**CommandCall**(*params*, *parsing=False*)

Bases: *alignak.alignakobject.AlignakObject*

This class is use when a service, contact or host define a command with args.

args

call

command

enable_environment_macros

get_command_and_args()

We want to get the command and the args with ! splitting. but don't forget to protect against the ! to avoid splitting on them

Remember: A Nagios-like command is command_name!arg1!arg2!...

Returns None

get_name()

Getter for call attribute

Returns call attribute

Return type str

is_valid()

Getter for valid attribute

Returns True if object is valid, False otherwise

Return type bool

late_relink_done

module_type

my_type = 'CommandCall'

poller_tag

properties = {'args': <Property <class 'alignak.property.ListProp'>, default: [] />,

reactionner_tag

serialize()

This function serializes into a simple dictionary object.

It is used when transferring data to other daemons over the network (http)

Here is the generic function that simply export attributes declared in the properties dictionary of the object.

Note that a SetProp property will be serialized as a list.

Returns Dictionary containing key and value from properties

Return type dict

timeout

valid

2.12 alignak.comment module

This module provide Comment class, used to attach comments to hosts / services

class alignak.comment.**Comment**(*params*, *parsing=False*)
Bases: *alignak.alignakobject.AlignakObject*

Comment class implements comments for monitoring purpose. It contains data like author, type etc..

get_comment_brok(*host_name*, *service_name=''*)
Get a comment brok

Parameters

- **host_name** –
- **service_name** –

Returns brok with wanted data

Return type *alignak.brok.Brok*

my_type = 'comment'

properties = {'author': <Property <class 'alignak.property.StringProp'>, default: u'

2.13 alignak.complexexpression module

This module provides ComplexExpressionNode and ComplexExpressionFactory used for parsing expression (business rules)

class alignak.complexexpression.**ComplexExpressionFactory**(*ctx='hostgroups'*, *grps=None*, *all_elements=None*)
Bases: object

ComplexExpressionFactory provides complex expression parsing functions

eval_cor_pattern(*pattern*)
Parse and build recursively a tree of ComplexExpressionNode from pattern

Parameters **pattern** (*str*) – pattern to parse

Returns root node of parsed tree

Type *alignak.complexexpression.ComplexExpressionNode*

find_object(*pattern*)
Get a list of host corresponding to the pattern regarding the context

Parameters **pattern** (*str*) – pattern to find

Returns Host list matching pattern (hostgroup name, template, all)

Return type list[*alignak.objects.host.Host*]

class alignak.complexexpression.**ComplexExpressionNode**
Bases: object

ComplexExpressionNode is a node class for complex_expression(s)

is_valid()
Check if all leaves are correct (no error)

Returns True if correct, else False

Return type bool

TODO: Fix this function and use it. DependencyNode should be ComplexExpressionNode Should return true on a leaf

resolve_elements()
Get element of this node recursively Compute rules with OR or AND rule then NOT rules.

Returns set of element

Return type set

2.14 alignak.contactdowntime module

This module provides ContactDowntime class which implement downtime for contact

class alignak.contactdowntime.**ContactDowntime**(*params*, *parsing=False*)
Bases: *alignak.alignakobject.AlignakObject*

ContactDowntime class allows a contact to be in downtime. During this time the contact won't get notifications

cancel(*contacts*)
Wrapper to call raise_cancel_downtime_log_entry for ref (host/service) set can_be_deleted to True set is_in_effect to False

Returns None

check_activation(*contacts*)
Enter or exit downtime if necessary

Returns None

enter(*contacts*)
Wrapper to call raise_enter_downtime_log_entry for ref (host/service)

Returns None

exit(*contacts*)
Wrapper to call raise_exit_downtime_log_entry for ref (host/service) set can_be_deleted to True

Returns None

in_scheduled_downtime()
Getter for is_in_effect attribute

Returns True if downtime is active, False otherwise

Return type bool

properties = {'author': <Property <class 'alignak.property.StringProp'>, default: u'

2.15 alignak.daemon module

This module provides abstraction for creating daemon in Alignak

class alignak.daemon.**Daemon**(*name*, ***kwargs*)

Bases: object

Class providing daemon level call for Alignak

add(*elt*)

Abstract method for adding brok, external commands, messages, ... It is overridden in subclasses (Satellite) of Daemon

Parameters **elt** – element to add

Returns None

change_to_user_group()

Change to configured user/group for the running program. If user/group are not valid, we exit with code 1 If change failed we exit with code 2

Returns None

change_to_workdir()

Change working directory to working attribute

Returns None

check_and_del_zombie_modules()

Check alive instance and try to restart the dead ones

Returns None

check_dir(*dirname*)

Check and create directory

Parameters **dirname** – file name

:type dirname; str

Returns None

check_for_system_time_change()

Check if our system time change. If so, change our

Returns 0 if the difference < 900, difference else

Return type int

check_parallel_run()

Check (in pid file) if there isn't already a daemon running. If yes and do_replace: kill it. Keep in self.fpid the File object to the pid file. Will be used by writepid.

Returns None

static check_shm()

Check /dev/shm right permissions

Returns None

close_fds(*skip_close_fds*)

Close all the process file descriptors. Skip the descriptors present in the skip_close_fds list

Parameters **skip_close_fds** (*list*) – list of file descriptor to preserve from closing

Returns None

compensate_system_time_change(*difference*)

Default action for system time change. Actually a log is done

Parameters **difference** (*int*) – in seconds

Returns None

daemon_connection_init(*s_link*, *set_wait_new_conf=False*)
Initialize a connection with the daemon for the provided satellite link

Initialize the connection (HTTP client) to the daemon and get its running identifier. Returns True if it succeeds else if any error occur or the daemon is inactive it returns False.

Assume the daemon should be reachable because we are initializing the connection... as such, force set the link reachable property

If set_wait_new_conf is set, the daemon is requested to wait a new configuration if we get a running identifier. This is used by the arbiter when a new configuration must be dispatched

NB: if the daemon is configured as passive, or if it is a daemon link that is inactive then it returns False without trying a connection.

Parameters

- **s_link** (SatelliteLink) – link of the daemon to connect to
- **set_wait_new_conf** (*bool*) – if the daemon must got the wait new configuration state

Returns True if the connection is established, else False

daemonize()
Go in "daemon" mode: close unused fds, redirect stdout/err, chdir, umask, fork-setsid-fork-writepid Do the double fork to properly go daemon

This is 'almost' as recommended by PEP3143 but it would be better to rewrite this daemonization thanks to the python-daemon library!

Returns None

do_before_loop()
Called before the main daemon loop.

Returns None

do_daemon_init_and_start(*set_proc_title=True*)
Main daemon function. Clean, allocates, initializes and starts all necessary resources to go in daemon mode.

The set_proc_title parameter is mainly useful for the Alignak unit tests. This to avoid changing the test process name!

Parameters **set_proc_title** (*bool*) – if set (default), the process title is changed to the daemon name

Returns False if the HTTP daemon can not be initialized, else True

do_load_modules(*modules*)
Wrapper for calling load_and_init method of modules_manager attribute

Parameters **modules** – list of modules that should be loaded by the daemon

Returns None

do_loop_turn()
Abstract method for daemon loop turn. It must be overridden by all classes inheriting from Daemon

Returns None

do_main_loop()
Main loop for an Alignak daemon

Returns None

do_stop()

Execute the stop of this daemon:

- request the daemon to stop
- request the http thread to stop, else force stop the thread
- Close the http socket
- Shutdown the manager
- Stop and join all started "modules"

Returns None

dump_environment()
Try to dump memory

Not currently implemented feature

Returns None

exit_ok(*message*, *exit_code=None*)
Log a message and exit

Parameters

- **exit_code** (*int*) – if not None, exit with the provided value as exit code
- **message** (*str*) – message for the exit reason

Returns None

exit_on_error(*message*, *exit_code=1*)
Log generic message when getting an error and exit

Parameters

- **exit_code** (*int*) – if not None, exit with the provided value as exit code
- **message** (*str*) – message for the exit reason

Returns None

exit_on_exception(*raised_exception*, *message=''*, *exit_code=99*)
Log generic message when getting an unrecoverable error

Parameters

- **raised_exception** (*Exception*) – raised Exception
- **message** (*str*) – message for the exit reason
- **exit_code** (*int*) – exit with the provided value as exit code

Returns None

get_daemon_stats(*details=False*)
Get state of modules and create a scheme for stats data of daemon This may be overridden in subclasses (and it is...)

Returns A dict with the following structure

::

{

'modules': { 'internal': {'name': "MYMODULE1", 'state': 'ok'}, 'external': {'name': "MYMODULE2", 'state': 'stopped'},

}, And some extra information, see the source code below...

}

These information are completed with the data provided by the get_id function which provides the daemon identification

Return type dict

get_header(*configuration=False*)
Get the log file header

If configuration is True, this returns the daemon configuration

Returns A string list containing project name, daemon name, version, licence etc.

Return type list

get_id(*details=False*)
Get daemon identification information

Returns A dict with the following structure

::

{ "alignak": selfAlignak instance name "type": daemon type "name": daemon name "version": Alignak version

}

Return type dict

get_links_of_type(*s_type=''*)
Return the *s_type* satellite list (eg. schedulers)

If s_type is None, returns a dictionary of all satellites, else returns the dictionary of the s_type satellites

The returned dict is indexed with the satellites uuid.

Parameters **s_type** (*str*) – satellite type

Returns dictionary of satellites

Return type dict

get_objects_from_from_queues()
Get objects from "from" queues and add them.

Returns True if we got something in the queue, False otherwise.

Return type bool

get_retention_data()
Basic function to get retention data, Maybe be overridden by subclasses to implement real get

TODO: only the scheduler is retention capable. To be removed!

Returns A list of Alignak object (scheduling items)

Return type list

hook_point(*hook_name*, *handle=None*)
Used to call module function that may define a hook function for hook_name

Available hook points: - *tick*, called on each daemon loop turn - *save_retention*; called by the scheduler when live state

saving is to be done

- ***load_retention*; called by the scheduler when live state** restoring is necessary (on restart)
- *get_new_actions*; called by the scheduler before adding the actions to be executed
- *early_configuration*; called by the arbiter when it begins parsing the configuration
- *read_configuration*; called by the arbiter when it read the configuration
- *late_configuration*; called by the arbiter when it finishes parsing the configuration

As a default, the *handle* parameter provided to the hooked function is the caller Daemon object. The scheduler will provide its own instance when it call this function.

Parameters

- **hook_name** (*str*) – function name we may hook in module
- **handle** – parameter to provide to the hook function

Type handle: alignak.Satellite

Returns None

http_daemon_thread()
Main function of the http daemon thread will loop forever unless we stop the root daemon

The main thing is to have a pool of X concurrent requests for the http_daemon, so "no_lock" calls can always be directly answer without having a "locked" version to finish. This is achieved thanks to the CherryPy thread pool.

This function is threaded to be detached from the main process as such it will not block the process main loop.. :return: None

load_modules_manager()
Instantiate the daemon ModulesManager and load the SyncManager (multiprocessing)

Note that this function is used by the Alignak modules. No self-use…

Parameters **daemon_name** – daemon name

Returns None

make_a_pause(*timeout=0.0001*, *check_time_change=True*)
Wait up to timeout and check for system time change.

This function checks if the system time changed since the last call. If so, the difference is returned to the caller. The duration of this call is removed from the timeout. If this duration is greater than the required timeout, no sleep is executed and the extra time is returned to the caller

If the required timeout was overlapped, then the first return value will be greater than the required timeout.

If the required timeout is null, then the timeout value is set as a very short time to keep a nice behavior to the system CPU ;)

Parameters

- **timeout** (*float*) – timeout to wait for activity
- **check_time_change** (*bool*) – True (default) to check if the system time changed

:return:Returns a 2-tuple: * first value is the time spent for the time change check * second value is the time change difference :rtype: tuple

manage_signal(*sig*, *frame*)
Manage signals caught by the daemon signal.SIGUSR1 : dump_environment signal.SIGUSR2 : dump_object (nothing) signal.SIGTERM, signal.SIGINT : terminate process

Parameters

- **sig** (*str*) – signal caught by daemon
- **frame** – current stack frame

Returns None

pidfile
Return the pid file name - make it compatible with old implementation

Returns pid_filename property

Return type str

properties = {'active': <Property <class 'alignak.property.BoolProp'>, default: True

request_stop(*message=''*, *exit_code=0*)
Remove pid and stop daemon

Returns None

restore_retention_data(*data*)
Basic function to save retention data, Maybe be overridden by subclasses to implement real save

TODO: only the scheduler is retention capable. To be removed!

Returns None

scheme
Daemon interface scheme

Returns http or https if the daemon uses SSL

Return type str

set_exit_handler(*sigs=None*)
Set the signal handler to manage_signal (defined in this class)

Only set handlers for: - signal.SIGTERM, signal.SIGINT - signal.SIGUSR1, signal.SIGUSR2 - signal.SIGHUP

Returns None

set_proctitle(*daemon_name=None*)
Set the proctitle of the daemon

Parameters **daemon_name** – daemon instance name (eg. arbiter-master). If not provided, only the

daemon type (eg. arbiter) will be used for the process title :type daemon_name: str :return: None

set_signal_handler(*sigs=None*)
Set the signal handler to manage_signal (defined in this class)

Only set handlers for: - signal.SIGTERM, signal.SIGINT - signal.SIGUSR1, signal.SIGUSR2 - signal.SIGHUP

Returns None

setup_alignak_logger()
Setup alignak logger: - with the daemon log configuration properties - configure the global daemon handler (root logger) - log the daemon Alignak header

- configure the global Alignak monitoring log

This function is called very early on daemon start. The daemon is not yet forked and may still run with a high privileged user account. This is why, the log file ownership may be set accordingly to the running user account.

Returns None

setup_communication_daemon()
Setup HTTP server daemon to listen for incoming HTTP requests from other Alignak daemons

Returns True if initialization is ok, else False

setup_new_conf()
Setup the new configuration received from Arbiter :return: None

unlink()
Remove the daemon's pid file

Returns None

wait_for_initial_conf(*timeout=1.0*)
Wait initial configuration from the arbiter. Basically sleep 1.0 and check if new_conf is here

Parameters **timeout** (*int*) – timeout to wait

Returns None

wait_for_new_conf(*timeout=1.0*)
Wait for a new configuration from the arbiter. Basically the same as waiting for an initial configuration (wait_for_initial_conf)

Parameters **timeout** (*int*) – timeout to wait

Returns None

watch_for_new_conf(*timeout=0*)
Check if a new configuration was sent to the daemon

This function is called on each daemon loop turn. Basically it is a sleep...

If a new configuration was posted, this function returns True

Parameters **timeout** (*float*) – timeout to wait. Default is no wait time.

Returns None

write_pid(*pid*)
Write pid to the pid file

Parameters **pid** (*None | int*) – pid of the process

Returns None

exception alignak.daemon.**EnvironmentFile**(*msg*)
Bases: exceptions.Exception

Exception raised when the Alignak environment file is missing or corrupted

alignak.daemon.**get_all_groups**()
Wrapper for getgrall

Returns all groups

Return type list

alignak.daemon.**get_cur_group**()
Wrapper for getgrgid

Returns group name

Return type str

alignak.daemon.**get_cur_user**()
Wrapper for getpwuid

Returns user name

Return type str

2.16 alignak.daterange module

This module provide Daterange and Timerange classes used to create Timeperiod in Alignak

class alignak.daterange.**AbstractDaterange**(*params=None*, *parsing=True*)
Bases: *alignak.alignakobject.AlignakObject*

AbstractDaterange class provides functions to deal with a range of dates It is subclassed for more granularity (weekday, month ...)

get_min_from_t(*timestamp*)
Get next time from t where a timerange is valid (withing range)

Parameters **timestamp** – base time to look for the next one

Returns time where a timerange is valid

Return type int

get_min_sec_from_morning()
Get the first second from midnight where a timerange is effective

Returns smallest amount of second from midnight of all timerange

Return type int

get_min_sec_out_from_morning()
Get the first second (from midnight) where we are out of a timerange

Returns smallest seconds from midnight of all timerange where it is not effective

Return type int

classmethod get_month_by_id(*month_id*)
Get month name from month id

Parameters **month_id** (*int*) – month id

Returns month name

Return type str

```
>>> Daterange.get_month_by_id(7)
'july'
```

classmethod get_month_id(*month*)
Get month id from month name

Parameters **month** (*str*) – month name

Returns month id

Return type int

```
>>> Daterange.get_month_id("july")
7
```

get_next_future_timerange_invalid(*timestamp*)
Get next invalid time for timeranges

Parameters **timestamp** (*int*) – time to check

Returns next time when a timerange is not valid

Return type None | int

get_next_future_timerange_valid(*timestamp*)
Get the next valid timerange (next timerange start in timeranges attribute)

Parameters **timestamp** (*int*) – base time

Returns next time when a timerange is valid

Return type None | int

get_next_invalid_day(*timestamp*)
Get next day where timerange is not active

Parameters **timestamp** (*int*) – time we compute from

Returns timestamp of the next invalid day (midnight) in LOCAL time.

Return type int | None

get_next_invalid_time_from_t(*timestamp*)
Get next invalid time for time range

Parameters **timestamp** (*int*) – time we compute from

Returns timestamp of the next invalid time (LOCAL TIME)

Return type int

get_next_valid_day(*timestamp*)
Get next valid day for timerange

Parameters **timestamp** (*int*) – time we compute from

Returns timestamp of the next valid day (midnight) in LOCAL time.

Return type int | None

get_next_valid_time_from_t(*timestamp*)
Get next valid time for time range

Parameters **timestamp** (*int*) – time we compute from

Returns timestamp of the next valid time (LOCAL TIME)

Return type int | None

get_start_and_end_time(*ref=None*)
Generic function to get start time and end time

Parameters **ref** (*int*) – time in seconds

Returns None

classmethod get_weekday_by_id(*weekday_id*)
Get weekday name from weekday id

Parameters **weekday_id** (*int*) – weekday id

Returns weekday name

Return type int

```
>>> Daterange.get_weekday_by_id(5)
'saturday'
```

classmethod get_weekday_id(*weekday*)
Get weekday id from weekday name

Parameters **weekday** (*str*) – weekday name

Returns weekday id

Return type int

```
>>> Daterange.get_weekday_id("monday")
0
```

is_correct()
Check if each timerange of this datarange is correct

Returns True if timerange are correct, False otherwise

Return type bool

is_time_day_invalid(*timestamp*)
Check if t is out of start time and end time of the DateRange

Parameters **timestamp** (*int*) – time to check

Returns False if t in range, True otherwise

Return type bool

is_time_day_valid(*timestamp*)
Check if it is within start time and end time of the DateRange

Parameters **timestamp** (*int*) – time to check

Returns True if t in range, False otherwise

Return type bool

is_time_valid(*timestamp*)
Check if time is valid for one of the timerange.

Parameters **timestamp** (*int*) – time to check

Returns True if one of the timerange is valid for t, False otherwise

Return type bool

timeranges = []

class alignak.daterange.**CalendarDaterange**(*params*, *parsing=True*)
Bases: *alignak.daterange.Daterange*

CalendarDaterange is for calendar entry (YYYY-MM-DD - YYYY-MM-DD)

get_start_and_end_time(*ref=None*)
Specific function to get start time and end time for CalendarDaterange

Parameters **ref** (*int*) – time in seconds

Returns tuple with start and end time

Return type tuple (int, int)

class alignak.daterange.**Daterange**(*params*, *parsing=True*)
Bases: *alignak.daterange.AbstractDaterange*

Daterange subclasses AbstractDaterange and instantiates Timerange objects

get_start_and_end_time(*ref=None*)
Generic function to get start time and end time

Parameters **ref** (*int*) – time in seconds

Returns None

months = {'april': 4, 'august': 8, 'december': 12, 'february': 2, 'january': 1, '

rev_months = {1: 'january', 2: 'february', 3: 'march', 4: 'april', 5: 'may', 6:

rev_weekdays = {0: 'monday', 1: 'tuesday', 2: 'wednesday', 3: 'thursday', 4: 'fri

serialize()
This function serialize into a simple dict object. It is used when transferring data to other daemons over the network (http)

Here we directly return all attributes

Returns json representation of a Daterange

Return type dict

weekdays = {'friday': 4, 'monday': 0, 'saturday': 5, 'sunday': 6, 'thursday': 3,

class alignak.daterange.**MonthDateDaterange**(*params*, *parsing=True*)
Bases: *alignak.daterange.Daterange*

MonthDateDaterange is for month and day entry (month DD - month DD)

get_start_and_end_time(*ref=None*)
Specific function to get start time and end time for MonthDateDaterange

Parameters **ref** (*int*) – time in seconds

Returns tuple with start and end time

Return type tuple (int, int)

class alignak.daterange.**MonthDayDaterange**(*params*, *parsing=True*)
Bases: *alignak.daterange.Daterange*

MonthDayDaterange is for month week day entry (day DD - DD)

get_start_and_end_time(*ref=None*)
Specific function to get start time and end time for MonthDayDaterange

Parameters **ref** (*int*) – time in seconds

Returns tuple with start and end time

Return type tuple (int, int)

class alignak.daterange.**MonthWeekDayDaterange**(*params*, *parsing=True*)
Bases: *alignak.daterange.Daterange*

MonthWeekDayDaterange is for month week day entry (weekday DD month - weekday DD month)

get_start_and_end_time(*ref=None*)
Specific function to get start time and end time for MonthWeekDayDaterange

Parameters **ref** (*int | None*) – time in seconds

Returns tuple with start and end time

Return type tuple

is_correct()
Check if the Daterange is correct : weekdays are valid

Returns True if weekdays are valid, False otherwise

Return type bool

class alignak.daterange.**StandardDaterange**(*params*, *parsing=True*)
Bases: *alignak.daterange.AbstractDaterange*

StandardDaterange is for standard entry (weekday - weekday)

get_start_and_end_time(*ref=None*)
Specific function to get start time and end time for StandardDaterange

Parameters **ref** (*int*) – time in seconds

Returns tuple with start and end time

Return type tuple (int, int)

is_correct()
Check if the Daterange is correct : weekdays are valid

Returns True if weekdays are valid, False otherwise

Return type bool

serialize()
This function serialize into a simple dict object. It is used when transferring data to other daemons over the network (http)

Here we directly return all attributes

Returns json representation of a Daterange

Return type dict

class alignak.daterange.**Timerange**(*entry=None*, *params=None*, *parsing=True*)
Bases: *alignak.alignakobject.AlignakObject*

Timerange class provides parsing facilities for time range declaration

get_first_sec_out_from_morning()
Get the first second (from midnight) where we are out of the timerange

Returns seconds from midnight where timerange is not effective

Return type int

get_sec_from_morning()
Get Timerange start time in seconds (from midnight)

Returns amount of seconds from midnight

Return type int

is_correct()
Getter for is_valid attribute

Returns True if Timerange is valid, False otherwise

Return type bool

is_time_valid(*timestamp*)
Check if time is valid for this Timerange

If sec_from_morning is not provided, get the value.

Parameters **timestamp** (*int*) – time to check

Returns True if time is valid (in interval), False otherwise

Return type bool

serialize()
This function serialize into a simple dict object. It is used when transferring data to other daemons over the network (http)

Here we directly return all attributes

Returns json representation of a Timerange

Return type dict

class alignak.daterange.**WeekDayDaterange**(*params*, *parsing=True*)
Bases: *alignak.daterange.Daterange*

WeekDayDaterange is for month week day entry (weekday offset - weekday offset)

get_start_and_end_time(*ref=None*)
Specific function to get start time and end time for WeekDayDaterange

Parameters **ref** (*int*) – time in seconds

Returns tuple with start and end time

Return type tuple (int, int)

alignak.daterange.**find_day_by_offset**(*year*, *month*, *offset*)
Get the month day based on date and offset

Parameters

- **year** (*int*) – date year
- **month** (*int*) – date month
- **offset** (*int*) – offset in day to compute (usually negative)

Returns day number in the month

Return type int

```
>>> find_day_by_offset(2015, 7, -1)
31
```

alignak.daterange.**find_day_by_weekday_offset**(*year*, *month*, *weekday*, *offset*)
Get the day number based on a date and offset

Parameters

- **year** (*int*) – date year
- **month** (*int*) – date month
- **weekday** (*int*) – date week day
- **offset** (*int*) – offset (-1 is last, 1 is first etc)

Returns day number in the month

Return type int

```
>>> find_day_by_weekday_offset(2010, 7, 1, -1)
27
```

alignak.daterange.**get_day**(*timestamp*)
Get timestamp of the beginning of the day (local) given by timestamp

Parameters **timestamp** (*int*) – time to get day from

Returns timestamp

Return type int

alignak.daterange.**get_end_of_day**(*year*, *month*, *day*)
Get the timestamp associated to the last second of a specific day

Parameters

- **year** (*int*) – date year
- **month** (*int*) – date month (int)
- **day** (*int*) – date day

Returns timestamp

Return type int

alignak.daterange.**get_sec_from_morning**(*timestamp*)
Get the number of seconds elapsed since the beginning of the day deducted from the provided timestamp

Parameters **timestamp** (*int*) – time to use for computation

Returns timestamp

Return type int

alignak.daterange.**get_start_of_day**(*year*, *month*, *day*)
Get the timestamp associated to the first second of a specific day

Parameters

- **year** (*int*) – date year
- **month** (*int*) – date month
- **day** (*int*) – date day

Returns timestamp

Return type int

alignak.daterange.**get_wday**(*timestamp*)
Get week day from date

Parameters **timestamp** (*int*) – timestamp date

Returns weekday (0-6)

Return type int

TODO: Not timezone aware

2.17 alignak.dependencynode module

This module provides DependencyNode and DependencyNodeFactory used for parsing expression (business rules)

class alignak.dependencynode.**DependencyNode**(*params=None*, *parsing=False*)
Bases: object

DependencyNode is a node class for business_rule expression(s)

get_complex_and_node_state(*hosts*, *services*)
Get state , handle AND aggregation

```
* Get the worst state. 2 or max of sons (3 <=> UNKNOWN < CRITICAL <=> 2)
* Revert if it's a not node
```

Parameters

- **hosts** – host objects
- **services** – service objects

Returns 0, 1 or 2

Return type int

get_complex_or_node_state(*hosts*, *services*)
Get state , handle OR aggregation

```
* Get the best state (min of sons)
* Revert if it's a not node
```

Parameters

- **hosts** – host objects
- **services** – service objects

Returns 0, 1 or 2

Return type int

get_complex_xof_node_state(*hosts*, *services*)
Get state , handle X of aggregation

```
* Count the number of OK, WARNING, CRITICAL
* Try too apply, in this order, Critical, Warning, OK rule
* Return the code for first match (2, 1, 0)
* If no rule apply, return OK for simple X of and worst state for multiple X␣
↪of
```

Parameters

- **hosts** – host objects
- **services** – service objects

Returns 0, 1 or 2

Return type int

TODO: Looks like the last if does the opposite of what the comment says

get_host_node_state(*state*, *problem_has_been_acknowledged*, *in_scheduled_downtime*)

Get host node state, simplest case

```
* Handle not value (revert) for host and consider 1 as 2
```

Returns 0, 1 or 2

Return type int

static get_reverse_state(*state*)

Do a symmetry around 1 of the state

```
* 0 -> 2
* 1 -> 1
* 2 -> 0
* else -> else
```

Parameters **state** (*int*) – state to reverse

Returns Integer from 0 to 2 (usually)

Return type int

get_service_node_state(*state*, *problem_has_been_acknowledged*, *in_scheduled_downtime*)

Get service node state, simplest case

```
* Handle not value (revert) for service
```

Returns 0, 1 or 2

Return type int

get_state(*hosts*, *services*)

Get node state by looking recursively over sons and applying operand

Parameters

- **hosts** – list of available hosts to search for
- **services** – list of available services to search for

Returns Node state

Return type int

is_valid()

Check if all leaves are correct (no error)

Returns True if correct, otherwise False

Return type bool

list_all_elements()
Get all host/service uuid in our node and below

Returns list of hosts/services uuids

Return type list

serialize()
This function serialize into a simple dict object. It is used when transferring data to other daemons over the network (http)

Here we directly return all attributes

Returns json representation of a DependencyNode

Return type dict

switch_zeros_of_values()

If we are a of: rule, we can get some 0 in of_values, if so, change them with NB sons instead

Returns None

class alignak.dependencynode.**DependencyNodeFactory**(*bound_item*)
Bases: object

DependencyNodeFactory provides dependency node parsing functions

eval_complex_cor_pattern(*pattern*, *hosts*, *services*, *hostgroups*, *servicegroups*, *running=False*)
Parse and build recursively a tree of DependencyNode from a complex pattern

Parameters

- **pattern** (*str*) – pattern to parse
- **hosts** (alignak.objects.host.Host) – hosts list, used to find a specific host
- **services** (alignak.objects.service.Service) – services list, used to find a specific service
- **running** (*bool*) – rules are evaluated at run time and parsing. True means runtime

Returns root node of parsed tree

Return type *alignak.dependencynode.DependencyNode*

eval_cor_pattern(*pattern*, *hosts*, *services*, *hostgroups*, *servicegroups*, *running=False*)
Parse and build recursively a tree of DependencyNode from pattern

Parameters

- **pattern** (*str*) – pattern to parse
- **hosts** (alignak.objects.host.Host) – hosts list, used to find a specific host
- **services** (alignak.objects.service.Service) – services list, used to find a specific service
- **running** (*bool*) – rules are evaluated at run time and parsing. True means runtime

Returns root node of parsed tree

Return type *alignak.dependencynode.DependencyNode*

eval_simple_cor_pattern (*pattern*, *hosts*, *services*, *hostgroups*, *servicegroups*, *running=False*)
Parse and build recursively a tree of DependencyNode from a simple pattern

Parameters

- **pattern** (*str*) – pattern to parse
- **hosts** (alignak.objects.host.Host) – hosts list, used to find a specific host
- **services** (alignak.objects.service.Service) – services list, used to find a specific service
- **running** (*bool*) – rules are evaluated at run time and parsing. True means runtime

Returns root node of parsed tree

Return type *alignak.dependencynode.DependencyNode*

static eval_xof_pattern (*node*, *pattern*)
Parse a X of pattern * Set is_of_mul attribute * Set of_values attribute

Parameters

- **node** – node to edit
- **pattern** (*str*) – line to match

Returns end of the line (without X of :)

Return type str

expand_expression (*pattern*, *hosts*, *services*, *hostgroups*, *servicegroups*, *running=False*)
Expand a host or service expression into a dependency node tree using (host|service)group membership, regex, or labels as item selector.

Parameters

- **pattern** (*str*) – pattern to parse
- **hosts** (alignak.objects.host.Host) – hosts list, used to find a specific host
- **services** (alignak.objects.service.Service) – services list, used to find a specific service
- **running** (*bool*) – rules are evaluated at run time and parsing. True means runtime

Returns root node of parsed tree

Return type *alignak.dependencynode.DependencyNode*

find_object (*pattern*, *hosts*, *services*)
Find object from pattern

Parameters

- **pattern** (*str*) – text to search (host1,service1)
- **hosts** (alignak.objects.host.Host) – hosts list, used to find a specific host
- **services** (alignak.objects.service.Service) – services list, used to find a specific service

Returns tuple with Host or Service object and error

Return type tuple

get_host_filters (*expr*)
Generates host filter list corresponding to the expression

```
* '*' => any
* 'g' => group filter
* 'r' => regex name filter
* 'l' => bp rule label filter
* 't' => tag  filter
* '' => none filter
* No flag match => host name filter
```

Parameters **expr** (*str*) – expression to parse

Returns filter list

Return type list

get_srv_host_filters(*expr*)

Generates service filter list corresponding to the expression

```
* '*' => any
* 'g' => hostgroup filter
* 'r' => host regex name filter
* 'l' => host bp rule label filter
* 't' => tag  filter
* '' => none filter
* No flag match => host name filter
```

Parameters **expr** (*str*) – expression to parse

Returns filter list

Return type list

get_srv_service_filters(*expr*)

Generates service filter list corresponding to the expression

```
* '*' => any
* 'g' => servicegroup filter
* 'r' => service regex name filter
* 'l' => service bp rule label filter
* 't' => tag  filter
* '' => none filter
* No flag match => service name filter
```

Parameters **expr** (*str*) – expression to parse

Returns filter list

Return type list

host_flags = 'grlt'

service_flags = 'grl'

2.18 alignak.dispatcher module

This is the Dispatcher class. Its role is to prepare the Alignak configuration to get dispatched to the Alignak satellites like schedulers, reactionners, pollers, receivers and brokers. It is responsible for high availability part. If an element

dies and the element type has a spare, it sends the config of the dead one to the spare one.

class alignak.dispatcher.**Dispatcher**(*conf*, *arbiter_link*)
Bases: object

Dispatcher is in charge of sending configuration to other daemon. It has to handle spare, realms, poller tags etc.

check_dispatch()
Check that all active satellites have a configuration dispatched

A DispatcherError exception is raised if no configuration is dispatched!

Returns None

check_reachable(*forced=False*, *test=False*)
Check all daemons state (reachable or not)

If test parameter is True, do not really send but simulate only for testing purpose...

The update_infos function returns None when no ping has been executed (too early...), or True / False according to the real ping and get managed configuration result. So, if the result is None, consider as not valid, else compute the global result...

Returns True if all daemons are reachable

check_status_and_get_events()
Get all the daemons status

Returns Dictionary with all the daemons returned information

Return type dict

dispatch(*test=False*)
Send configuration to satellites

Returns None

get_satellites_list(*sat_type*)
Get a sorted satellite list: master then spare

Parameters **sat_type** (*str*) – type of the required satellites (arbiters, schedulers, ...)

Returns sorted satellites list

Return type list[*alignak.objects.satellitelink.SatelliteLink*]

get_scheduler_ordered_list(*realm*)
Get sorted scheduler list for a specific realm

List is ordered as: alive first, then spare (if any), then dead scheduler links

Parameters **realm** (alignak.objects.realm.Realm) – realm we want scheduler from

Returns sorted scheduler list

Return type list[*alignak.objects.schedulerlink.SchedulerLink*]

prepare_dispatch()
Prepare dispatch, so prepare for each daemon (schedulers, brokers, receivers, reactionners, pollers)

This function will only prepare something if self.new_to_dispatch is False It will reset the first_dispatch_done flag

A DispatcherError exception is raised if a configuration is already prepared! Unset the new_to_dispatch flag before calling!

Returns None

stop_request(*stop_now=False*)
Send a stop request to all the daemons

Parameters **stop_now** (*bool*) – stop now or go to stop wait mode

Returns True if all daemons are reachable

exception alignak.dispatcher.**DispatcherError**(*msg*)
Bases: exceptions.Exception

Exception raised for errors in the configuration dispatching.

Attributes: msg – explanation of the error

2.19 alignak.downtime module

This modules provides Downtime class, used to implements downtime monitoring concept. See detailed concepts below

class alignak.downtime.**Downtime**(*params*, *parsing=False*)
Bases: *alignak.alignakobject.AlignakObject*

Schedules downtime for a specified service. If the "fixed" argument is set to one (1), downtime will start and end at the times specified by the "start" and "end" arguments. Otherwise, downtime will begin between the "start" and "end" times and last for "duration" seconds. The "start" and "end" arguments are specified in time_t format (seconds since the UNIX epoch). The specified service downtime can be triggered by another downtime entry if the "trigger_id" is set to the ID of another scheduled downtime entry. Set the "trigger_id" argument to zero (0) if the downtime for the specified service should not be triggered by another downtime entry.

add_automatic_comment(*ref*)
Add comment on ref for downtime

Parameters **ref** (alignak.objects.schedulingitem.SchedulingItem) – the host/service we want to link a comment to

Returns None

cancel(*timeperiods*, *hosts*, *services*)
Remove ref in scheduled downtime and raise downtime log entry (cancel)

Parameters

- **hosts** (alignak.objects.host.Hosts) – hosts objects to get item ref
- **services** (alignak.objects.service.Services) – services objects to get item ref

Returns [], always

Return type list

del_automatic_comment(*item*)
Remove automatic comment on ref previously created

Parameters **item** (*object*) – item service or host

Returns None

enter(*timeperiods*, *hosts*, *services*)
Set ref in scheduled downtime and raise downtime log entry (start)

Parameters

- **hosts** (alignak.objects.host.Hosts) – hosts objects to get item ref
- **services** (alignak.objects.service.Services) – services objects to get item ref

Returns broks

Return type list of broks

exit(*timeperiods*, *hosts*, *services*)
Remove ref in scheduled downtime and raise downtime log entry (exit)

Parameters

- **hosts** (alignak.objects.host.Hosts) – hosts objects to get item ref
- **services** (alignak.objects.service.Services) – services objects to get item ref

Returns [], always | None

Return type list

fill_data_brok_from(*data*, *brok_type*)
Fill data with info of item by looking at brok_type in props of properties or running_properties

Parameters

- **data** – data to fill
- **brok_type** (*str*) – type of brok

Returns None

TODO: Duplicate from Notification.fill_data_brok_from

get_expire_brok(*host_name*, *service_name=''*)
Get an expire downtime brok

Parameters **host_name** – host concerned by the downtime

:type host_name :param service_name: service concerned by the downtime :type service_name :return: brok with wanted data :rtype: alignak.brok.Brok

get_raise_brok(*host_name*, *service_name=''*)
Get a start downtime brok

Parameters **host_name** – host concerned by the downtime

:type host_name :param service_name: service concerned by the downtime :type service_name :return: brok with wanted data :rtype: alignak.brok.Brok

in_scheduled_downtime()
Getter for is_in_effect attribute

Returns True if downtime is in effect, False otherwise

Return type bool

my_type = 'downtime'

properties = {'activate_me': <Property <class 'alignak.property.StringProp'>, default

trigger_me(*other_downtime*)
Wrapper to activate_me.append function Used to add another downtime to activate

Parameters **other_downtime** – other downtime to activate/cancel

Returns None

2.20 alignak.eventhandler module

This module provides EventHandler class, used when hosts or services reach a bad state.

class alignak.eventhandler.**EventHandler**(*params=None*, *parsing=False*)

Bases: *alignak.action.Action*

Notification class, inherits from action class. Used to execute action when a host or a service is in a bad state

check_time

command

creation_time

env

execution_time

exit_status

get_outputs(*out*, *max_plugins_output_length*)

Setter of output attribute

Parameters

- **out** (*str*) – new output
- **max_plugins_output_length** (*int*) – not used

Returns None

get_return_from(*e_handler*)

Setter of the following attributes:

```
* exit_status
* output
* long_output
* check_time
* execution_time
* perf_data
```

Parameters **e_handler** (alignak.eventhandler.EventHandler) – event handler to get data from

Returns None

internal

is_a

is_snapshot

last_poll

long_output

module_type

my_scheduler

my_type = 'eventhandler'

my_worker

output

perf_data

properties = {'_in_timeout': <Property <class 'alignak.property.BoolProp'>, default:

reactionner_tag

ref

ref_type

s_time

status

t_to_go

timeout

type

u_time

wait_time

2.21 alignak.external_command module

This module provides ExternalCommand and ExternalCommandManager classes Used to process command sent by users

class alignak.external_command.**ExternalCommand**(*cmd_line*, *timestamp=None*)

Bases: object

ExternalCommand class is only an object with a cmd_line attribute. All parsing and execution is done in manager

my_type = 'externalcommand'

serialize()

This function serializes into a simple dict object. It is used when transferring data to other daemons over the network (http)

Here we directly return all attributes

Returns json representation of a Brok

Return type dict

class alignak.external_command.**ExternalCommandManager**(*conf*, *mode*, *daemon*, *accept_unknown=False*, *log_external_commands=False*)

Bases: object

ExternalCommandManager manages all external commands sent to Alignak.

It basically parses arguments and executes the right function

acknowledge_host_problem(*host*, *sticky*, *notify*, *author*, *comment*)
Acknowledge a host problem Format of the line that triggers function call:

ACKNOWLEDGE_HOST_PROBLEM;<host_name>;<sticky>;<notify>;<persistent:obsolete>;<author>;<comment>

Parameters

- **host** (alignak.objects.host.Host) – host to acknowledge the problem
- **sticky** – if sticky == 2, the acknowledge will remain until the host returns to an

UP state else the acknowledge will be removed as soon as the host state changes :type sticky: integer :param notify: if to 1, send a notification :type notify: integer :param author: name of the author or the acknowledge :type author: str :param comment: comment (description) of the acknowledge :type comment: str :return: None TODO: add a better ACK management

acknowledge_host_problem_expire(*host*, *sticky*, *notify*, *end_time*, *author*, *comment*)
Acknowledge a host problem with expire time for this acknowledgement Format of the line that triggers function call:

ACKNOWLEDGE_HOST_PROBLEM_EXPIRE;<host_name>;<sticky>;<notify>;<persistent:obsolete>;<end_time>;<author>;<comment>

Parameters

- **host** (alignak.objects.host.Host) – host to acknowledge the problem
- **sticky** (*integer*) – acknowledge will be always present is host return in UP state
- **notify** (*integer*) – if to 1, send a notification
- **end_time** (*int*) – end (timeout) of this acknowledge in seconds(timestamp) (0 to never end)
- **author** (*str*) – name of the author or the acknowledge
- **comment** (*str*) – comment (description) of the acknowledge

Returns None

TODO: add a better ACK management

acknowledge_svc_problem(*service*, *sticky*, *notify*, *author*, *comment*)
Acknowledge a service problem Format of the line that triggers function call:

ACKNOWLEDGE_SVC_PROBLEM;<host_name>;<service_description>;<sticky>;<notify>;<persistent:obsolete>;<author>;<comment>

Parameters

- **service** (alignak.objects.service.Service) – service to acknowledge the problem
- **sticky** – if sticky == 2, the acknowledge will remain until the service returns to an

OK state else the acknowledge will be removed as soon as the service state changes :param notify: if to 1, send a notification :type notify: integer :param author: name of the author or the acknowledge :type author: str :param comment: comment (description) of the acknowledge :type comment: str :return: None

acknowledge_svc_problem_expire(*service*, *sticky*, *notify*, *end_time*, *author*, *comment*)
Acknowledge a service problem with expire time for this acknowledgement Format of the line that triggers function call:

ACKNOWLEDGE_SVC_PROBLEM_EXPIRE;<host_name>;<service_description>;<sticky>;<notify>;<persistent:obsolete>;<end_time>;<author>;<comment>

Parameters

- **service** (alignak.objects.service.Service) – service to acknowledge the problem
- **sticky** (*integer*) – acknowledge will be always present is host return in UP state
- **notify** (*integer*) – if to 1, send a notification
- **end_time** (*int*) – end (timeout) of this acknowledge in seconds(timestamp) (0 to never end)
- **author** (*str*) – name of the author or the acknowledge
- **comment** (*str*) – comment (description) of the acknowledge

Returns None

add_host_comment(*host*, *author*, *comment*)

Add a host comment Format of the line that triggers function call:

ADD_HOST_COMMENT;<host_name>;<persistent:obsolete>;<author>;<comment>

Parameters

- **host** (alignak.objects.host.Host) – host to add the comment
- **author** (*str*) – author name
- **comment** (*str*) – text comment

Returns None

add_svc_comment(*service*, *author*, *comment*)

Add a service comment Format of the line that triggers function call:

ADD_SVC_COMMENT;<host_name>;<service_description>;<persistent:obsolete>;<author>;<comment>

Parameters

- **service** (alignak.objects.service.Service) – service to add the comment
- **author** (*str*) – author name
- **comment** (*str*) – text comment

Returns None

change_contact_host_notification_timeperiod(*contact*, *notification_timeperiod*)

Change contact host notification timeperiod value Format of the line that triggers function call:

CHANGE_CONTACT_HOST_NOTIFICATION_TIMEPERIOD;<contact_name>;<notification_timeperiod>

Parameters

- **contact** (alignak.objects.contact.Contact) – contact to edit
- **notification_timeperiod** (alignak.objects.timeperiod.Timeperiod) – timeperiod to set

Returns None

static change_contact_modattr(*contact*, *value*)

Change contact modified attribute value Format of the line that triggers function call:

CHANGE_CONTACT_MODATTR;<contact_name>;<value>

Parameters

- **contact** (alignak.objects.contact.Contact) – contact to edit
- **value** (*str*) – new value to set

Returns None

static change_contact_modhattr(*contact*, *value*)
Change contact modified host attribute value Format of the line that triggers function call:

CHANGE_CONTACT_MODHATTR;<contact_name>;<value>

Parameters

- **contact** (alignak.objects.contact.Contact) – contact to edit
- **value** – new value to set

:type value:str :return: None

static change_contact_modsattr(*contact*, *value*)
Change contact modified service attribute value Format of the line that triggers function call:

CHANGE_CONTACT_MODSATTR;<contact_name>;<value>

Parameters

- **contact** (alignak.objects.contact.Contact) – contact to edit
- **value** (*str*) – new value to set

Returns None

change_contact_svc_notification_timeperiod(*contact*, *notification_timeperiod*)
Change contact service notification timeperiod value Format of the line that triggers function call:

CHANGE_CONTACT_SVC_NOTIFICATION_TIMEPERIOD;<contact_name>;<notification_timeperiod>

Parameters

- **contact** (alignak.objects.contact.Contact) – contact to edit
- **notification_timeperiod** (alignak.objects.timeperiod.Timeperiod) – timeperiod to set

Returns None

change_custom_contact_var(*contact*, *varname*, *varvalue*)
Change custom contact variable Format of the line that triggers function call:

CHANGE_CUSTOM_CONTACT_VAR;<contact_name>;<varname>;<varvalue>

Parameters

- **contact** (alignak.objects.contact.Contact) – contact to edit
- **varname** (*str*) – variable name to change
- **varvalue** (*str*) – variable new value

Returns None

change_custom_host_var(*host*, *varname*, *varvalue*)
Change custom host variable Format of the line that triggers function call:

CHANGE_CUSTOM_HOST_VAR;<host_name>;<varname>;<varvalue>

Parameters

- **host** (alignak.objects.host.Host) – host to edit

- **varname** (*str*) – variable name to change
- **varvalue** (*str*) – variable new value

Returns None

change_custom_svc_var(*service*, *varname*, *varvalue*)
Change custom service variable Format of the line that triggers function call:

CHANGE_CUSTOM_SVC_VAR;<host_name>;<service_description>;<varname>;<varvalue>

Parameters

- **service** (alignak.objects.service.Service) – service to edit
- **varname** (*str*) – variable name to change
- **varvalue** (*str*) – variable new value

Returns None

change_global_host_event_handler(*event_handler_command*)
DOES NOTHING (should change global host event handler) Format of the line that triggers function call:

CHANGE_GLOBAL_HOST_EVENT_HANDLER;<event_handler_command>

Parameters **event_handler_command** – new event handler

Returns None

TODO: DICT_MODATTR["MODATTR_EVENT_HANDLER_COMMAND"].value

change_global_svc_event_handler(*event_handler_command*)
DOES NOTHING (should change global service event handler) Format of the line that triggers function call:

CHANGE_GLOBAL_SVC_EVENT_HANDLER;<event_handler_command>

Parameters **event_handler_command** – new event handler

Returns None

TODO: DICT_MODATTR["MODATTR_EVENT_HANDLER_COMMAND"].value

change_host_check_command(*host*, *check_command*)
Modify host check command Format of the line that triggers function call:

CHANGE_HOST_CHECK_COMMAND;<host_name>;<check_command>

Parameters

- **host** (alignak.objects.host.Host) – host to modify check command
- **check_command** – command line

Returns None

change_host_check_timeperiod(*host*, *timeperiod*)
Modify host check timeperiod Format of the line that triggers function call:

CHANGE_HOST_CHECK_TIMEPERIOD;<host_name>;<timeperiod>

Parameters

- **host** (alignak.objects.host.Host) – host to modify check timeperiod
- **timeperiod** (alignak.objects.timeperiod.Timeperiod) – timeperiod object

Returns None

change_host_event_handler(*host*, *event_handler_command*)
Modify host event handler Format of the line that triggers function call:

CHANGE_HOST_EVENT_HANDLER;<host_name>;<event_handler_command>

Parameters

- **host** (alignak.objects.host.Host) – host to modify event handler
- **event_handler_command** – event handler command line

Returns None

change_host_modattr(*host*, *value*)
Change host modified attributes Format of the line that triggers function call:

CHANGE_HOST_MODATTR;<host_name>;<value>

For boolean attributes, toggles the service attribute state (enable/disable) For non boolean attribute, only indicates that the corresponding attribute is to be saved in the retention.

Value can be: MODATTR_NONE 0 MODATTR_NOTIFICATIONS_ENABLED 1 MODATTR_ACTIVE_CHECKS_ENABLED 2 MODATTR_PASSIVE_CHECKS_ENABLED 4 MODATTR_EVENT_HANDLER_ENABLED 8 MODATTR_FLAP_DETECTION_ENABLED 16 MODATTR_PERFORMANCE_DATA_ENABLED 64 MODATTR_EVENT_HANDLER_COMMAND 256 MODATTR_CHECK_COMMAND 512 MODATTR_NORMAL_CHECK_INTERVAL 1024 MODATTR_RETRY_CHECK_INTERVAL 2048 MODATTR_MAX_CHECK_ATTEMPTS 4096 MODATTR_FRESHNESS_CHECKS_ENABLED 8192 MODATTR_CHECK_TIMEPERIOD 16384 MODATTR_CUSTOM_VARIABLE 32768 MODATTR_NOTIFICATION_TIMEPERIOD 65536

Parameters

- **host** (alignak.objects.host.Host) – host to edit
- **value** (*str*) – new value to set

Returns None

change_host_snapshot_command(*host*, *snapshot_command*)
Modify host snapshot command Format of the line that triggers function call:

CHANGE_HOST_SNAPSHOT_COMMAND;<host_name>;<event_handler_command>

Parameters

- **host** (alignak.objects.host.Host) – host to modify snapshot command
- **snapshot_command** – snapshot command command line

Returns None

change_max_host_check_attempts(*host*, *check_attempts*)
Modify max host check attempt Format of the line that triggers function call:

CHANGE_MAX_HOST_CHECK_ATTEMPTS;<host_name>;<check_attempts>

Parameters

- **host** (alignak.objects.host.Host) – host to edit
- **check_attempts** (*int*) – new value to set

Returns None

change_max_svc_check_attempts(*service*, *check_attempts*)
Modify max service check attempt Format of the line that triggers function call:

CHANGE_MAX_SVC_CHECK_ATTEMPTS;<host_name>;<service_description>;<check_attempts>

Parameters

- **service** (alignak.objects.service.Service) – service to edit
- **check_attempts** (*int*) – new value to set

Returns None

change_normal_host_check_interval(*host*, *check_interval*)
Modify host check interval Format of the line that triggers function call:

CHANGE_NORMAL_HOST_CHECK_INTERVAL;<host_name>;<check_interval>

Parameters

- **host** (alignak.objects.host.Host) – host to edit
- **check_interval** – new value to set

Returns None

change_normal_svc_check_interval(*service*, *check_interval*)
Modify service check interval Format of the line that triggers function call:

CHANGE_NORMAL_SVC_CHECK_INTERVAL;<host_name>;<service_description>;<check_interval>

Parameters

- **service** (alignak.objects.service.Service) – service to edit
- **check_interval** – new value to set

Returns None

change_retry_host_check_interval(*host*, *check_interval*)
Modify host retry interval Format of the line that triggers function call:

CHANGE_RETRY_HOST_CHECK_INTERVAL;<host_name>;<check_interval>

Parameters

- **host** (alignak.objects.host.Host) – host to edit
- **check_interval** – new value to set

Returns None

change_retry_svc_check_interval(*service*, *check_interval*)
Modify service retry interval Format of the line that triggers function call:

CHANGE_RETRY_SVC_CHECK_INTERVAL;<host_name>;<service_description>;<check_interval>

Parameters

- **service** (alignak.objects.service.Service) – service to edit
- **check_interval** – new value to set

Returns None

change_svc_check_command(*service*, *check_command*)
Modify service check command Format of the line that triggers function call:

CHANGE_SVC_CHECK_COMMAND;<host_name>;<service_description>;<check_command>

Parameters

- **service** (alignak.objects.service.Service) – service to modify check command
- **check_command** – command line

Returns None

change_svc_check_timeperiod(*service*, *check_timeperiod*)
Modify service check timeperiod Format of the line that triggers function call:

CHANGE_SVC_CHECK_TIMEPERIOD;<host_name>;<service_description>;<check_timeperiod>

Parameters

- **service** (alignak.objects.service.Service) – service to modify check timeperiod
- **check_timeperiod** (alignak.objects.timeperiod.Timeperiod) – timeperiod object

Returns None

change_svc_event_handler(*service*, *event_handler_command*)
Modify service event handler Format of the line that triggers function call:

CHANGE_SVC_EVENT_HANDLER;<host_name>;<service_description>;<event_handler_command>

Parameters

- **service** (alignak.objects.service.Service) – service to modify event handler
- **event_handler_command** – event handler command line

Returns None

change_svc_modattr(*service*, *value*)
Change service modified attributes Format of the line that triggers function call:

CHANGE_SVC_MODATTR;<host_name>;<service_description>;<value>

For boolean attributes, toggles the service attribute state (enable/disable) For non boolean attribute, only indicates that the corresponding attribute is to be saved in the retention.

Value can be: MODATTR_NONE 0 MODATTR_NOTIFICATIONS_ENABLED 1 MODATTR_ACTIVE_CHECKS_ENABLED 2 MODATTR_PASSIVE_CHECKS_ENABLED 4 MODATTR_EVENT_HANDLER_ENABLED 8 MODATTR_FLAP_DETECTION_ENABLED 16 MODATTR_PERFORMANCE_DATA_ENABLED 64 MODATTR_EVENT_HANDLER_COMMAND 256 MODATTR_CHECK_COMMAND 512 MODATTR_NORMAL_CHECK_INTERVAL 1024 MODATTR_RETRY_CHECK_INTERVAL 2048 MODATTR_MAX_CHECK_ATTEMPTS 4096 MODATTR_FRESHNESS_CHECKS_ENABLED 8192 MODATTR_CHECK_TIMEPERIOD 16384 MODATTR_CUSTOM_VARIABLE 32768 MODATTR_NOTIFICATION_TIMEPERIOD 65536

Parameters

- **service** (alignak.objects.service.Service) – service to edit
- **value** (*str*) – new value to set / unset

Returns None

change_svc_notification_timeperiod(*service*, *notification_timeperiod*)
Change service notification timeperiod Format of the line that triggers function call:

CHANGE_SVC_NOTIFICATION_TIMEPERIOD;<host_name>;<service_description>; <notification_timeperiod>

Parameters

- **service** (alignak.objects.service.Service) – service to edit
- **notification_timeperiod** (alignak.objects.timeperiod.Timeperiod) – timeperiod to set

Returns None

change_svc_snapshot_command(*service*, *snapshot_command*)

Modify host snapshot command Format of the line that triggers function call:

CHANGE_HOST_SNAPSHOT_COMMAND;<host_name>;<event_handler_command>

Parameters

- **service** (alignak.objects.service.Service) – service to modify snapshot command
- **snapshot_command** – snapshot command command line

Returns None

commands = {'acknowledge_host_problem': {'args': ['host', 'to_int', 'to_bool', 'obso

del_all_contact_downtimes(*contact*)

Delete all contact downtimes Format of the line that triggers function call:

DEL_ALL_CONTACT_DOWNTIMES;<contact_name>

Parameters **contact** (alignak.objects.contact.Contact) – contact to edit

Returns None

del_all_host_comments(*host*)

Delete all host comments Format of the line that triggers function call:

DEL_ALL_HOST_COMMENTS;<host_name>

Parameters **host** (alignak.objects.host.Host) – host to edit

Returns None

del_all_host_downtimes(*host*)

Delete all host downtimes Format of the line that triggers function call:

DEL_ALL_HOST_DOWNTIMES;<host_name>

Parameters **host** (alignak.objects.host.Host) – host to edit

Returns None

del_all_svc_comments(*service*)

Delete all service comments Format of the line that triggers function call:

DEL_ALL_SVC_COMMENTS;<host_name>;<service_description>

Parameters **service** (alignak.objects.service.Service) – service to edit

Returns None

del_all_svc_downtimes(*service*)

Delete all service downtime Format of the line that triggers function call:

DEL_ALL_SVC_DOWNTIMES;<host_name>;<service_description>

Parameters **service** (alignak.objects.service.Service) – service to edit

Returns None

del_contact_downtime (*downtime_id*)
Delete a contact downtime Format of the line that triggers function call:

DEL_CONTACT_DOWNTIME;<downtime_id>

Parameters **downtime_id** (*int*) – downtime id to delete

Returns None

del_host_comment (*comment_id*)
Delete a host comment Format of the line that triggers function call:

DEL_HOST_COMMENT;<comment_id>

Parameters **comment_id** (*int*) – comment id to delete

Returns None

del_host_downtime (*downtime_id*)
Delete a host downtime Format of the line that triggers function call:

DEL_HOST_DOWNTIME;<downtime_id>

Parameters **downtime_id** (*int*) – downtime id to delete

Returns None

del_svc_comment (*comment_id*)
Delete a service comment Format of the line that triggers function call:

DEL_SVC_COMMENT;<comment_id>

Parameters **comment_id** (*int*) – comment id to delete

Returns None

del_svc_downtime (*downtime_id*)
Delete a service downtime Format of the line that triggers function call:

DEL_SVC_DOWNTIME;<downtime_id>

Parameters **downtime_id** (*int*) – downtime id to delete

Returns None

delay_host_notification (*host*, *notification_time*)
Modify host first notification delay Format of the line that triggers function call:

DELAY_HOST_NOTIFICATION;<host_name>;<notification_time>

Parameters

- **host** (alignak.objects.host.Host) – host to edit
- **notification_time** – new value to set

Returns None

delay_svc_notification (*service*, *notification_time*)
Modify service first notification delay Format of the line that triggers function call:

DELAY_SVC_NOTIFICATION;<host_name>;<service_description>;<notification_time>

Parameters

- **service** (alignak.objects.service.Service) – service to edit
- **notification_time** – new value to set

Returns None

disable_all_notifications_beyond_host(*host*)
DOES NOTHING (should disable notification beyond a host) Format of the line that triggers function call:

DISABLE_ALL_NOTIFICATIONS_BEYOND_HOST;<host_name>

Parameters **host** (alignak.objects.host.Host) – host to edit

Returns None

TODO: Implement it

disable_contact_host_notifications(*contact*)
Disable host notifications for a contact Format of the line that triggers function call:

DISABLE_CONTACT_HOST_NOTIFICATIONS;<contact_name>

Parameters **contact** (alignak.objects.contact.Contact) – contact to disable

Returns None

disable_contact_svc_notifications(*contact*)
Disable service notifications for a contact Format of the line that triggers function call:

DISABLE_CONTACT_SVC_NOTIFICATIONS;<contact_name>

Parameters **contact** (alignak.objects.contact.Contact) – contact to disable

Returns None

disable_contactgroup_host_notifications(*contactgroup*)
Disable host notifications for a contactgroup Format of the line that triggers function call:

DISABLE_CONTACTGROUP_HOST_NOTIFICATIONS;<contactgroup_name>

Parameters **contactgroup** (alignak.objects.contactgroup.Contactgroup) – contactgroup to disable

Returns None

disable_contactgroup_svc_notifications(*contactgroup*)
Disable service notifications for a contactgroup Format of the line that triggers function call:

DISABLE_CONTACTGROUP_SVC_NOTIFICATIONS;<contactgroup_name>

Parameters **contactgroup** (alignak.objects.contactgroup.Contactgroup) – contactgroup to disable

Returns None

disable_event_handlers()
Disable event handlers (globally) Format of the line that triggers function call:

DISABLE_EVENT_HANDLERS

Returns None

disable_flap_detection()
Disable flap detection (globally) Format of the line that triggers function call:

DISABLE_FLAP_DETECTION

Returns None

disable_host_and_child_notifications(*host*)
DOES NOTHING (Should disable host notifications and its child) Format of the line that triggers function call:

DISABLE_HOST_AND_CHILD_NOTIFICATIONS;<host_name

Parameters **host** (alignak.objects.host.Host) – host to edit

Returns None

disable_host_check(*host*)
Disable checks for a host Format of the line that triggers function call:

DISABLE_HOST_CHECK;<host_name>

Parameters **host** (alignak.objects.host.Host) – host to edit

Returns None

disable_host_event_handler(*host*)
Disable event handlers for a host Format of the line that triggers function call:

DISABLE_HOST_EVENT_HANDLER;<host_name>

Parameters **host** (alignak.objects.host.Host) – host to edit

Returns None

disable_host_flap_detection(*host*)
Disable flap detection for a host Format of the line that triggers function call:

DISABLE_HOST_FLAP_DETECTION;<host_name>

Parameters **host** (alignak.objects.host.Host) – host to edit

Returns None

disable_host_freshness_check(*host*)
Disable freshness check for a host Format of the line that triggers function call:

DISABLE_HOST_FRESHNESS_CHECK;<host_name>

Parameters **host** (alignak.objects.host.Host) – host to edit

Returns None

disable_host_freshness_checks()
Disable freshness checks (globally) Format of the line that triggers function call:

DISABLE_HOST_FRESHNESS_CHECKS

Returns None

disable_host_notifications(*host*)
Disable notifications for a host Format of the line that triggers function call:

DISABLE_HOST_NOTIFICATIONS;<host_name>

Parameters **host** (alignak.objects.host.Host) – host to edit

Returns None

disable_host_svc_checks(*host*)
Disable service checks for a host Format of the line that triggers function call:

DISABLE_HOST_SVC_CHECKS;<host_name>

Parameters **host** (alignak.objects.host.Host) – host to edit

Returns None

disable_host_svc_notifications(*host*)
Disable services notifications for a host Format of the line that triggers function call:

DISABLE_HOST_SVC_NOTIFICATIONS;<host_name>

Parameters **host** (alignak.objects.host.Host) – host to edit

Returns None

disable_hostgroup_host_checks(*hostgroup*)
Disable host checks for a hostgroup Format of the line that triggers function call:

DISABLE_HOSTGROUP_HOST_CHECKS;<hostgroup_name>

Parameters **hostgroup** (alignak.objects.hostgroup.Hostgroup) – hostgroup to disable

Returns None

disable_hostgroup_host_notifications(*hostgroup*)
Disable host notifications for a hostgroup Format of the line that triggers function call:

DISABLE_HOSTGROUP_HOST_NOTIFICATIONS;<hostgroup_name>

Parameters **hostgroup** (alignak.objects.hostgroup.Hostgroup) – hostgroup to disable

Returns None

disable_hostgroup_passive_host_checks(*hostgroup*)
Disable host passive checks for a hostgroup Format of the line that triggers function call:

DISABLE_HOSTGROUP_PASSIVE_HOST_CHECKS;<hostgroup_name>

Parameters **hostgroup** (alignak.objects.hostgroup.Hostgroup) – hostgroup to disable

Returns None

disable_hostgroup_passive_svc_checks(*hostgroup*)
Disable service passive checks for a hostgroup Format of the line that triggers function call:

DISABLE_HOSTGROUP_PASSIVE_SVC_CHECKS;<hostgroup_name>

Parameters **hostgroup** (alignak.objects.hostgroup.Hostgroup) – hostgroup to disable

Returns None

disable_hostgroup_svc_checks(*hostgroup*)
Disable service checks for a hostgroup Format of the line that triggers function call:

DISABLE_HOSTGROUP_SVC_CHECKS;<hostgroup_name>

Parameters **hostgroup** (alignak.objects.hostgroup.Hostgroup) – hostgroup to disable

Returns None

disable_hostgroup_svc_notifications(*hostgroup*)
Disable service notifications for a hostgroup Format of the line that triggers function call:

DISABLE_HOSTGROUP_SVC_NOTIFICATIONS;<hostgroup_name>

Parameters **hostgroup** (alignak.objects.hostgroup.Hostgroup) – hostgroup to disable

Returns None

disable_notifications()
Disable notifications (globally) Format of the line that triggers function call:

DISABLE_NOTIFICATIONS

Returns None

disable_passive_host_checks(*host*)
Disable passive checks for a host Format of the line that triggers function call:

DISABLE_PASSIVE_HOST_CHECKS;<host_name>

Parameters **host** (alignak.objects.host.Host) – host to edit

Returns None

disable_passive_svc_checks(*service*)
Disable passive checks for a service Format of the line that triggers function call:

DISABLE_PASSIVE_SVC_CHECKS;<host_name>;<service_description>

Parameters **service** (alignak.objects.service.Service) – service to edit

Returns None

disable_performance_data()
Disable performance data processing (globally) Format of the line that triggers function call:

DISABLE_PERFORMANCE_DATA

Returns None

disable_service_flap_detection(*service*)
Disable flap detection for a service Format of the line that triggers function call:

DISABLE_SERVICE_FLAP_DETECTION;<host_name>;<service_description>

Parameters **service** (alignak.objects.service.Service) – service to edit

Returns None

disable_service_freshness_checks()
Disable service freshness checks (globally) Format of the line that triggers function call:

DISABLE_SERVICE_FRESHNESS_CHECKS

Returns None

disable_servicegroup_host_checks(*servicegroup*)
Disable host checks for a servicegroup Format of the line that triggers function call:

DISABLE_SERVICEGROUP_HOST_CHECKS;<servicegroup_name>

Parameters **servicegroup** (alignak.objects.servicegroup.Servicegroup) – servicegroup to disable

Returns None

disable_servicegroup_host_notifications(*servicegroup*)
Disable host notifications for a servicegroup Format of the line that triggers function call:

DISABLE_SERVICEGROUP_HOST_NOTIFICATIONS;<servicegroup_name>

Parameters **servicegroup** (alignak.objects.servicegroup.Servicegroup) – servicegroup to disable

Returns None

disable_servicegroup_passive_host_checks(*servicegroup*)
Disable passive host checks for a servicegroup Format of the line that triggers function call:

DISABLE_SERVICEGROUP_PASSIVE_HOST_CHECKS;<servicegroup_name>

Parameters **servicegroup** (alignak.objects.servicegroup.Servicegroup) – servicegroup to disable

Returns None

disable_servicegroup_passive_svc_checks(*servicegroup*)
Disable passive service checks for a servicegroup Format of the line that triggers function call:

DISABLE_SERVICEGROUP_PASSIVE_SVC_CHECKS;<servicegroup_name>

Parameters **servicegroup** (alignak.objects.servicegroup.Servicegroup) – servicegroup to disable

Returns None

disable_servicegroup_svc_checks(*servicegroup*)
Disable service checks for a servicegroup Format of the line that triggers function call:

DISABLE_SERVICEGROUP_SVC_CHECKS;<servicegroup_name>

Parameters **servicegroup** (alignak.objects.servicegroup.Servicegroup) – servicegroup to disable

Returns None

disable_servicegroup_svc_notifications(*servicegroup*)
Disable service notifications for a servicegroup Format of the line that triggers function call:

DISABLE_SERVICEGROUP_SVC_NOTIFICATIONS;<servicegroup_name>

Parameters **servicegroup** (alignak.objects.servicegroup.Servicegroup) – servicegroup to disable

Returns None

disable_svc_check(*service*)
Disable checks for a service Format of the line that triggers function call:

DISABLE_SVC_CHECK;<host_name>;<service_description>

Parameters **service** (alignak.objects.service.Service) – service to edit

Returns None

disable_svc_event_handler(*service*)
Disable event handlers for a service Format of the line that triggers function call:

DISABLE_SVC_EVENT_HANDLER;<host_name>;<service_description>

Parameters **service** (alignak.objects.service.Service) – service to edit

Returns None

disable_svc_flap_detection(*service*)
Disable flap detection for a service Format of the line that triggers function call:

DISABLE_SVC_FLAP_DETECTION;<host_name>;<service_description>

Parameters **service** (alignak.objects.service.Service) – service to edit

Returns None

disable_svc_freshness_check(*service*)
Disable freshness check for a service Format of the line that triggers function call:

DISABLE_SERVICE_FRESHNESS_CHECK;<host_name>;<service_description>

Parameters **service** (alignak.objects.service.Service) – service to edit

Returns None

disable_svc_notifications(*service*)
Disable notifications for a service Format of the line that triggers function call:

DISABLE_SVC_NOTIFICATIONS;<host_name>;<service_description>

Parameters **service** (alignak.objects.service.Service) – service to edit

Returns None

enable_all_notifications_beyond_host(*host*)
DOES NOTHING (should enable notification beyond a host) Format of the line that triggers function call:

ENABLE_ALL_NOTIFICATIONS_BEYOND_HOST;<host_name>

Parameters **host** (alignak.objects.host.Host) – host to edit

Returns None

TODO: Implement it

enable_contact_host_notifications(*contact*)
Enable host notifications for a contact Format of the line that triggers function call:

ENABLE_CONTACT_HOST_NOTIFICATIONS;<contact_name>

Parameters **contact** (alignak.objects.contact.Contact) – contact to enable

Returns None

enable_contact_svc_notifications(*contact*)
Enable service notifications for a contact Format of the line that triggers function call:

DISABLE_CONTACT_SVC_NOTIFICATIONS;<contact_name>

Parameters **contact** (alignak.objects.contact.Contact) – contact to enable

Returns None

enable_contactgroup_host_notifications(*contactgroup*)
Enable host notifications for a contactgroup Format of the line that triggers function call:

ENABLE_CONTACTGROUP_HOST_NOTIFICATIONS;<contactgroup_name>

Parameters **contactgroup** (alignak.objects.contactgroup.Contactgroup) – contactgroup to enable

Returns None

enable_contactgroup_svc_notifications(*contactgroup*)
Enable service notifications for a contactgroup Format of the line that triggers function call:

ENABLE_CONTACTGROUP_SVC_NOTIFICATIONS;<contactgroup_name>

Parameters **contactgroup** (alignak.objects.contactgroup.Contactgroup) – contactgroup to enable

Returns None

enable_event_handlers()
Enable event handlers (globally) Format of the line that triggers function call:

ENABLE_EVENT_HANDLERS

Returns None

enable_flap_detection()
Enable flap detection (globally) Format of the line that triggers function call:

ENABLE_FLAP_DETECTION

Returns None

enable_host_and_child_notifications(*host*)
DOES NOTHING (Should enable host notifications and its child) Format of the line that triggers function call:

ENABLE_HOST_AND_CHILD_NOTIFICATIONS;<host_name>

Parameters **host** (alignak.objects.host.Host) – host to edit

Returns None

enable_host_check(*host*)
Enable checks for a host Format of the line that triggers function call:

ENABLE_HOST_CHECK;<host_name>

Parameters **host** (alignak.objects.host.Host) – host to edit

Returns None

enable_host_event_handler(*host*)
Enable event handlers for a host Format of the line that triggers function call:

ENABLE_HOST_EVENT_HANDLER;<host_name>

Parameters **host** (alignak.objects.host.Host) – host to edit

Returns None

enable_host_flap_detection(*host*)
Enable flap detection for a host Format of the line that triggers function call:

ENABLE_HOST_FLAP_DETECTION;<host_name>

Parameters **host** (alignak.objects.host.Host) – host to edit

Returns None

enable_host_freshness_check(*host*)
Enable freshness check for a host Format of the line that triggers function call:

ENABLE_HOST_FRESHNESS_CHECK;<host_name>

Parameters **host** (alignak.objects.host.Host) – host to edit

Returns None

enable_host_freshness_checks()
Enable freshness checks (globally) Format of the line that triggers function call:

ENABLE_HOST_FRESHNESS_CHECKS

Returns None

enable_host_notifications(*host*)
Enable notifications for a host Format of the line that triggers function call:

ENABLE_HOST_NOTIFICATIONS;<host_name>

Parameters **host** (alignak.objects.host.Host) – host to edit

Returns None

enable_host_svc_checks(*host*)
Enable service checks for a host Format of the line that triggers function call:

ENABLE_HOST_SVC_CHECKS;<host_name>

Parameters **host** (alignak.objects.host.Host) – host to edit

Returns None

enable_host_svc_notifications(*host*)
Enable services notifications for a host Format of the line that triggers function call:

ENABLE_HOST_SVC_NOTIFICATIONS;<host_name>

Parameters **host** (alignak.objects.host.Host) – host to edit

Returns None

enable_hostgroup_host_checks(*hostgroup*)
Enable host checks for a hostgroup Format of the line that triggers function call:

ENABLE_HOSTGROUP_HOST_CHECKS;<hostgroup_name>

Parameters **hostgroup** (alignak.objects.hostgroup.Hostgroup) – hostgroup to enable

Returns None

enable_hostgroup_host_notifications(*hostgroup*)
Enable host notifications for a hostgroup Format of the line that triggers function call:

ENABLE_HOSTGROUP_HOST_NOTIFICATIONS;<hostgroup_name>

Parameters **hostgroup** (alignak.objects.hostgroup.Hostgroup) – hostgroup to enable

Returns None

enable_hostgroup_passive_host_checks(*hostgroup*)
Enable host passive checks for a hostgroup Format of the line that triggers function call:

ENABLE_HOSTGROUP_PASSIVE_HOST_CHECKS;<hostgroup_name>

Parameters **hostgroup** (alignak.objects.hostgroup.Hostgroup) – hostgroup to enable

Returns None

enable_hostgroup_passive_svc_checks(*hostgroup*)
Enable service passive checks for a hostgroup Format of the line that triggers function call:

ENABLE_HOSTGROUP_PASSIVE_SVC_CHECKS;<hostgroup_name>

Parameters **hostgroup** (alignak.objects.hostgroup.Hostgroup) – hostgroup to enable

Returns None

enable_hostgroup_svc_checks(*hostgroup*)
Enable service checks for a hostgroup Format of the line that triggers function call:

ENABLE_HOSTGROUP_SVC_CHECKS;<hostgroup_name>

Parameters **hostgroup** (alignak.objects.hostgroup.Hostgroup) – hostgroup to enable

Returns None

enable_hostgroup_svc_notifications(*hostgroup*)
Enable service notifications for a hostgroup Format of the line that triggers function call:

ENABLE_HOSTGROUP_SVC_NOTIFICATIONS;<hostgroup_name>

Parameters **hostgroup** (alignak.objects.hostgroup.Hostgroup) – hostgroup to enable

Returns None

enable_notifications()
Enable notifications (globally) Format of the line that triggers function call:

ENABLE_NOTIFICATIONS

Returns None

enable_passive_host_checks(*host*)
Enable passive checks for a host Format of the line that triggers function call:

ENABLE_PASSIVE_HOST_CHECKS;<host_name>

Parameters **host** (alignak.objects.host.Host) – host to edit

Returns None

enable_passive_svc_checks(*service*)
Enable passive checks for a service Format of the line that triggers function call:

ENABLE_PASSIVE_SVC_CHECKS;<host_name>;<service_description>

Parameters **service** (alignak.objects.service.Service) – service to edit

Returns None

enable_performance_data()
Enable performance data processing (globally) Format of the line that triggers function call:

ENABLE_PERFORMANCE_DATA

Returns None

enable_service_freshness_checks()
Enable service freshness checks (globally) Format of the line that triggers function call:

ENABLE_SERVICE_FRESHNESS_CHECKS

Returns None

enable_servicegroup_host_checks(*servicegroup*)
Enable host checks for a servicegroup Format of the line that triggers function call:

ENABLE_SERVICEGROUP_HOST_CHECKS;<servicegroup_name>

Parameters **servicegroup** (alignak.objects.servicegroup.Servicegroup) – servicegroup to enable

Returns None

enable_servicegroup_host_notifications(*servicegroup*)
Enable host notifications for a servicegroup Format of the line that triggers function call:

ENABLE_SERVICEGROUP_HOST_NOTIFICATIONS;<servicegroup_name>

Parameters **servicegroup** (alignak.objects.servicegroup.Servicegroup) – servicegroup to enable

Returns None

enable_servicegroup_passive_host_checks(*servicegroup*)
Enable passive host checks for a servicegroup Format of the line that triggers function call:

ENABLE_SERVICEGROUP_PASSIVE_HOST_CHECKS;<servicegroup_name>

Parameters **servicegroup** (alignak.objects.servicegroup.Servicegroup) – servicegroup to enable

Returns None

enable_servicegroup_passive_svc_checks(*servicegroup*)
Enable passive service checks for a servicegroup Format of the line that triggers function call:

ENABLE_SERVICEGROUP_PASSIVE_SVC_CHECKS;<servicegroup_name>

Parameters **servicegroup** (alignak.objects.servicegroup.Servicegroup) – servicegroup to enable

Returns None

enable_servicegroup_svc_checks(*servicegroup*)
Enable service checks for a servicegroup Format of the line that triggers function call:

ENABLE_SERVICEGROUP_SVC_CHECKS;<servicegroup_name>

Parameters **servicegroup** (alignak.objects.servicegroup.Servicegroup) – servicegroup to enable

Returns None

enable_servicegroup_svc_notifications(*servicegroup*)
Enable service notifications for a servicegroup Format of the line that triggers function call:

ENABLE_SERVICEGROUP_SVC_NOTIFICATIONS;<servicegroup_name>

Parameters **servicegroup** (alignak.objects.servicegroup.Servicegroup) – servicegroup to enable

Returns None

enable_svc_check(*service*)
Enable checks for a service Format of the line that triggers function call:

ENABLE_SVC_CHECK;<host_name>;<service_description>

Parameters **service** (alignak.objects.service.Service) – service to edit

Returns None

enable_svc_event_handler(*service*)
Enable event handlers for a service Format of the line that triggers function call:

ENABLE_SVC_EVENT_HANDLER;<host_name>;<service_description>

Parameters **service** (alignak.objects.service.Service) – service to edit

Returns None

enable_svc_flap_detection(*service*)
Enable flap detection for a service Format of the line that triggers function call:

ENABLE_SVC_FLAP_DETECTION;<host_name>;<service_description>

Parameters **service** (alignak.objects.service.Service) – service to edit

Returns None

enable_svc_freshness_check(*service*)
Enable freshness check for a service Format of the line that triggers function call:

ENABLE_SERVICE_FRESHNESS_CHECK;<host_name>;<service_description>

Parameters **service** (alignak.objects.service.Service) – service to edit

Returns None

enable_svc_notifications(*service*)
Enable notifications for a service Format of the line that triggers function call:

ENABLE_SVC_NOTIFICATIONS;<host_name>;<service_description>

Parameters **service** (alignak.objects.service.Service) – service to edit

Returns None

get_command_and_args(*command*, *extcmd=None*)
Parse command and get args

Parameters

- **command** (*str*) – command line to parse
- **extcmd** (*None | object*) – external command object (used to dispatch)

Returns Dict containing command and arg

{'global': False, 'c_name': c_name, 'args': args}

Return type dict | None

static get_unknown_check_result_brok(*cmd_line*)
Create unknown check result brok and fill it with command data

Parameters **cmd_line** (*str*) – command line to extract data

Returns unknown check result brok

Return type alignak.objects.brok.Brok

launch_host_event_handler(*host*)
Launch event handler for a service Format of the line that triggers function call:

LAUNCH_HOST_EVENT_HANDLER;<host_name>

Parameters **host** (alignak.objects.host.Host) – host to execute the event handler

Returns None

launch_svc_event_handler(*service*)
Launch event handler for a service Format of the line that triggers function call:

LAUNCH_SVC_EVENT_HANDLER;<host_name>;<service_description>

Parameters **service** (alignak.objects.service.Service) – service to execute the event handler

Returns None

process_file(*file_name*, *delete*)
DOES NOTHING (should process a file) Format of the line that triggers function call:

PROCESS_FILE;<file_name>;<delete>

Parameters

- **file_name** (*str*) – file to process
- **delete** – delete after processing

Returns None

process_host_check_result(*host*, *status_code*, *plugin_output*)
Process host check result Format of the line that triggers function call:

PROCESS_HOST_CHECK_RESULT;<host_name>;<status_code>;<plugin_output>

Parameters

- **host** (alignak.objects.host.Host) – host to process check to
- **status_code** (*int*) – exit code of plugin
- **plugin_output** (*str*) – plugin output

Returns None

TODO: say that check is PASSIVE

process_host_output(*host*, *plugin_output*)
Process host output Format of the line that triggers function call:

PROCESS_HOST_OUTPUT;<host_name>;<plugin_output>

Parameters

- **host** (alignak.objects.host.Host) – host to process check to
- **plugin_output** (*str*) – plugin output

Returns None

process_service_check_result(*service*, *return_code*, *plugin_output*)
Process service check result Format of the line that triggers function call:

PROCESS_SERVICE_CHECK_RESULT;<host_name>;<service_description>;<return_code>;<plugin_output>

Parameters

- **service** (alignak.objects.service.Service) – service to process check to
- **return_code** (*int*) – exit code of plugin
- **plugin_output** (*str*) – plugin output

Returns None

process_service_output(*service*, *plugin_output*)
Process service output Format of the line that triggers function call:

PROCESS_SERVICE_OUTPUT;<host_name>;<service_description>;<plugin_output>

Parameters

- **service** (alignak.objects.service.Service) – service to process check to

- **plugin_output** (*str*) – plugin output

Returns None

read_state_information()
Request to load the live state from the retention storage Format of the line that triggers function call:

READ_STATE_INFORMATION

Returns None

reload_config()
Reload Alignak configuration Format of the line that triggers function call:

RELOAD_CONFIG

Returns None

static remove_host_acknowledgement(*host*)
Remove an acknowledgment on a host Format of the line that triggers function call:

REMOVE_HOST_ACKNOWLEDGEMENT;<host_name>

Parameters **host** (alignak.objects.host.Host) – host to edit

Returns None

static remove_svc_acknowledgement(*service*)
Remove an acknowledgment on a service Format of the line that triggers function call:

REMOVE_SVC_ACKNOWLEDGEMENT;<host_name>;<service_description>

Parameters **service** (alignak.objects.service.Service) – service to edit

Returns None

resolve_command(*excmd*)
Parse command and dispatch it (to schedulers for example) if necessary If the command is not global it will be executed.

Parameters **excmd** (alignak.external_command.ExternalCommand) – external command to handle

Returns result of command parsing. None for an invalid command.

restart_program()
Restart Alignak Format of the line that triggers function call:

RESTART_PROGRAM

Returns None

save_state_information()
Request to save the live state to the retention Format of the line that triggers function call:

SAVE_STATE_INFORMATION

Returns None

schedule_and_propagate_host_downtime(*host*, *start_time*, *end_time*, *fixed*, *trigger_id*, *duration*, *author*, *comment*)
DOES NOTHING (Should create host downtime and start it?) Format of the line that triggers function call:

SCHEDULE_AND_PROPAGATE_HOST_DOWNTIME;<host_name>;<start_time>;<end_time>;<fixed>;<trigger_id>;<duration>;<author>;<comment>

Returns None

schedule_and_propagate_triggered_host_downtime(*host*, *start_time*, *end_time*, *fixed*, *trigger_id*, *duration*, *author*, *comment*)
DOES NOTHING (Should create triggered host downtime and start it?) Format of the line that triggers function call:

SCHEDULE_AND_PROPAGATE_TRIGGERED_HOST_DOWNTIME;<host_name>;<start_time>;<end_time>;<fixed>;<trigger_id>;<duration>;<author>;<comment>

Returns None

schedule_contact_downtime(*contact*, *start_time*, *end_time*, *author*, *comment*)
Schedule contact downtime Format of the line that triggers function call:

SCHEDULE_CONTACT_DOWNTIME;<contact_name>;<start_time>;<end_time>;<author>;<comment>

Parameters

- **contact** (*alignak.objects.contact.Contact*) – contact to put in downtime
- **start_time** (*int*) – downtime start time
- **end_time** (*int*) – downtime end time
- **author** (*str*) – downtime author
- **comment** (*str*) – text comment

Returns None

schedule_forced_host_check(*host*, *check_time*)
Schedule a forced check on a host Format of the line that triggers function call:

SCHEDULE_FORCED_HOST_CHECK;<host_name>;<check_time>

Parameters

- **host** (*alignak.object.host.Host*) – host to check
- **check_time** (*int*) – time to check

Returns None

schedule_forced_host_svc_checks(*host*, *check_time*)
Schedule a forced check on all services of a host Format of the line that triggers function call:

SCHEDULE_FORCED_HOST_SVC_CHECKS;<host_name>;<check_time>

Parameters

- **host** (*alignak.object.host.Host*) – host to check
- **check_time** (*int*) – time to check

Returns None

schedule_forced_svc_check(*service*, *check_time*)
Schedule a forced check on a service Format of the line that triggers function call:

SCHEDULE_FORCED_SVC_CHECK;<host_name>;<service_description>;<check_time>

Parameters

- **service** (*alignak.object.service.Service*) – service to check
- **check_time** (*int*) – time to check

Returns None

schedule_host_check(*host*, *check_time*)
Schedule a check on a host Format of the line that triggers function call:

SCHEDULE_HOST_CHECK;<host_name>;<check_time>

Parameters

- **host** (*alignak.object.host.Host*) – host to check
- **check_time** – time to check

Returns None

schedule_host_downtime(*host*, *start_time*, *end_time*, *fixed*, *trigger_id*, *duration*, *author*, *comment*)
Schedule a host downtime Format of the line that triggers function call:

SCHEDULE_HOST_DOWNTIME;<host_name>;<start_time>;<end_time>;<fixed>; <trigger_id>;<duration>;<author>;<comment>

Parameters

- **host** (*alignak.object.host.Host*) – host to schedule downtime
- **start_time** – downtime start time
- **end_time** – downtime end time
- **fixed** (*bool*) – is downtime fixed
- **trigger_id** (*str*) – downtime id that triggered this one
- **duration** (*int*) – downtime duration
- **author** (*str*) – downtime author
- **comment** (*str*) – downtime comment

Returns None

schedule_host_svc_checks(*host*, *check_time*)
Schedule a check on all services of a host Format of the line that triggers function call:

SCHEDULE_HOST_SVC_CHECKS;<host_name>;<check_time>

Parameters

- **host** (*alignak.object.host.Host*) – host to check
- **check_time** – time to check

Returns None

schedule_host_svc_downtime(*host*, *start_time*, *end_time*, *fixed*, *trigger_id*, *duration*, *author*, *comment*)
Schedule a service downtime for each service of an host Format of the line that triggers function call:

SCHEDULE_HOST_SVC_DOWNTIME;<host_name>;<start_time>;<end_time>;
<fixed>;<trigger_id>;<duration>;<author>;<comment>

Parameters

- **host** (*alignak.object.host.Host*) – host to schedule downtime
- **start_time** – downtime start time
- **end_time** – downtime end time
- **fixed** (*bool*) – is downtime fixed

- **trigger_id** (*str*) – downtime id that triggered this one
- **duration** (*int*) – downtime duration
- **author** (*str*) – downtime author
- **comment** (*str*) – downtime comment

Returns None

schedule_hostgroup_host_downtime(*hostgroup*, *start_time*, *end_time*, *fixed*, *trigger_id*, *duration*, *author*, *comment*)

Schedule a downtime for each host of a hostgroup Format of the line that triggers function call:

SCHEDULE_HOSTGROUP_HOST_DOWNTIME;<hostgroup_name>;<start_time>;<end_time>;<fixed>;<trigger_id>;<duration>;<author>;<comment>

Parameters

- **hostgroup** (alignak.objects.hostgroup.Hostgroup) – hostgroup to schedule
- **start_time** – downtime start time
- **end_time** – downtime end time
- **fixed** – is downtime fixed
- **trigger_id** (*str*) – downtime id that triggered this one
- **duration** (*int*) – downtime duration
- **author** (*str*) – downtime author
- **comment** (*str*) – downtime comment

Returns None

schedule_hostgroup_svc_downtime(*hostgroup*, *start_time*, *end_time*, *fixed*, *trigger_id*, *duration*, *author*, *comment*)

Schedule a downtime for each service of each host of a hostgroup Format of the line that triggers function call:

SCHEDULE_HOSTGROUP_SVC_DOWNTIME;;<hostgroup_name>;<start_time>;<end_time>;<fixed>;<trigger_id>;<duration>;<author>;<comment>

Parameters

- **hostgroup** (alignak.objects.hostgroup.Hostgroup) – hostgroup to schedule
- **start_time** – downtime start time
- **end_time** – downtime end time
- **fixed** – is downtime fixed
- **trigger_id** (*str*) – downtime id that triggered this one
- **duration** (*int*) – downtime duration
- **author** (*str*) – downtime author
- **comment** (*str*) – downtime comment

Returns None

schedule_servicegroup_host_downtime(*servicegroup*, *start_time*, *end_time*, *fixed*, *trigger_id*, *duration*, *author*, *comment*)

Schedule a host downtime for each host of services in a servicegroup Format of the line that triggers function call:

SCHEDULE_SERVICEGROUP_HOST_DOWNTIME;<servicegroup_name>;<start_time>;<end_time>;<fixed>;<trigger_id>;<duration>;<author>;<comment>

Parameters

- **servicegroup** (*alignak.object.servicegroup.Servicegroup*) – servicegroup to schedule downtime
- **start_time** – downtime start time
- **end_time** – downtime end time
- **fixed** (*bool*) – is downtime fixed
- **trigger_id** (*str*) – downtime id that triggered this one
- **duration** (*int*) – downtime duration
- **author** (*str*) – downtime author
- **comment** (*str*) – downtime comment

Returns None

schedule_servicegroup_svc_downtime(*servicegroup*, *start_time*, *end_time*, *fixed*, *trigger_id*, *duration*, *author*, *comment*)

Schedule a service downtime for each service of a servicegroup Format of the line that triggers function call:

SCHEDULE_SERVICEGROUP_SVC_DOWNTIME;<servicegroup_name>;<start_time>;<end_time>;<fixed>;<trigger_id>;<duration>;<author>;<comment>

Parameters

- **servicegroup** (*alignak.object.servicegroup.Servicegroup*) – servicegroup to schedule downtime
- **start_time** – downtime start time
- **end_time** – downtime end time
- **fixed** (*bool*) – is downtime fixed
- **trigger_id** (*str*) – downtime id that triggered this one
- **duration** (*int*) – downtime duration
- **author** (*str*) – downtime author
- **comment** (*str*) – downtime comment

Returns None

schedule_svc_check(*service*, *check_time*)

Schedule a check on a service Format of the line that triggers function call:

SCHEDULE_SVC_CHECK;<host_name>;<service_description>;<check_time>

Parameters

- **service** (*alignak.object.service.Service*) – service to check
- **check_time** – time to check

Returns None

schedule_svc_downtime(*service*, *start_time*, *end_time*, *fixed*, *trigger_id*, *duration*, *author*, *comment*)

Schedule a service downtime Format of the line that triggers function call:

SCHEDULE_SVC_DOWNTIME;<host_name>;<service_description><start_time>;<end_time>;<fixed>;<trigger_id>;<duration>;<author>;<comment>

Parameters

- **service** (*alignak.object.service.Service*) – service to check
- **start_time** – downtime start time
- **end_time** – downtime end time
- **fixed** (*bool*) – is downtime fixed
- **trigger_id** (*int*) – downtime id that triggered this one
- **duration** (*int*) – downtime duration
- **author** (*str*) – downtime author
- **comment** (*str*) – downtime comment

Returns None

search_host_and_dispatch(*host_name*, *command*, *extcmd*)

Try to dispatch a command for a specific host (so specific scheduler) because this command is related to a host (change notification interval for example)

Parameters

- **host_name** (*str*) – host name to search
- **command** (*str*) – command line
- **extcmd** (alignak.external_command.ExternalCommand) – external command object (the object will be added to sched commands list)

Returns None

send_an_element(*element*)

Send an element (Brok, Comment,...) to our daemon

Use the daemon *add* function if it exists, else raise an error log

Parameters **element** – elementto be sent

Type alignak.Brok, or *Comment*, or *Downtime*, ..

Returns

send_custom_host_notification(*host*, *options*, *author*, *comment*)

DOES NOTHING (Should send a custom notification) Format of the line that triggers function call:

SEND_CUSTOM_HOST_NOTIFICATION;<host_name>;<options>;<author>;<comment>

Parameters

- **host** (*alignak.object.host.Host*) – host to send notif for
- **options** – notification options
- **author** (*str*) – notification author
- **comment** (*str*) – notification text

Returns None

send_custom_svc_notification(*service*, *options*, *author*, *comment*)
DOES NOTHING (Should send a custom notification) Format of the line that triggers function call:

SEND_CUSTOM_SVC_NOTIFICATION;<host_name>;<service_description>;<options>;<author>;<comment>>

Parameters

- **service** (*alignak.object.service.Service*) – service to send notif for
- **options** – notification options
- **author** (*str*) – notification author
- **comment** (*str*) – notification text

Returns None

set_host_notification_number(*host*, *notification_number*)
DOES NOTHING (Should set host notification number) Format of the line that triggers function call:

SET_HOST_NOTIFICATION_NUMBER;<host_name>;<notification_number>

Parameters

- **host** (*alignak.object.host.Host*) – host to edit
- **notification_number** – new value to set

Returns None

set_svc_notification_number(*service*, *notification_number*)
DOES NOTHING (Should set host notification number) Format of the line that triggers function call:

SET_SVC_NOTIFICATION_NUMBER;<host_name>;<service_description>;<notification_number>

Parameters

- **service** (*alignak.object.service.Service*) – service to edit
- **notification_number** – new value to set

Returns None

shutdown_program()
DOES NOTHING (Should shutdown Alignak) Format of the line that triggers function call:

SHUTDOWN_PROGRAM

Returns None

start_accepting_passive_host_checks()
Enable passive host check submission (globally) Format of the line that triggers function call:

START_ACCEPTING_PASSIVE_HOST_CHECKS

Returns None

start_accepting_passive_svc_checks()
Enable passive service check submission (globally) Format of the line that triggers function call:

START_ACCEPTING_PASSIVE_SVC_CHECKS

Returns None

start_executing_host_checks()
Enable host check execution (globally) Format of the line that triggers function call:

START_EXECUTING_HOST_CHECKS

Returns None

start_executing_svc_checks()
Enable service check execution (globally) Format of the line that triggers function call:

START_EXECUTING_SVC_CHECKS

Returns None

stop_accepting_passive_host_checks()
Disable passive host check submission (globally) Format of the line that triggers function call:

STOP_ACCEPTING_PASSIVE_HOST_CHECKS

Returns None

stop_accepting_passive_svc_checks()
Disable passive service check submission (globally) Format of the line that triggers function call:

STOP_ACCEPTING_PASSIVE_SVC_CHECKS

Returns None

stop_executing_host_checks()
Disable host check execution (globally) Format of the line that triggers function call:

STOP_EXECUTING_HOST_CHECKS

Returns None

stop_executing_svc_checks()
Disable service check execution (globally) Format of the line that triggers function call:

STOP_EXECUTING_SVC_CHECKS

Returns None

2.22 alignak.graph module

This modules provide Graph class. Used to check for loop into dependencies

class alignak.graph.**Graph**
Bases: object

Graph is a class to make graph things like DFS checks or accessibility Why use an atomic bomb when a little hammer is enough? Graph are oriented.

add_edge(*from_node*, *to_node*)
Add edge between two node The edge is oriented

Parameters

- **from_node** (*object*) – node where edge starts
- **to_node** (*object*) – node where edge ends

Returns None

add_node(*node*)
Create the node key into the mode dict with [] value

Parameters **node** (*object*) – node to add

Returns None

add_nodes(*nodes*)
Add several nodes into the nodes dict

Parameters **nodes** (*object*) – nodes to add

Returns None

dfs_get_all_childs(*root*)
Recursively get all sons of this node

Parameters **root** – node to get sons

Returns sons

Return type list

dfs_loop_search(*root*)
Main algorithm to look for loop. It tags nodes and find ones stuck in loop.

- Init all nodes with DFS_UNCHECKED value
- DFS_TEMPORARY_CHECKED means we found it once
- DFS_OK : this node (and all sons) are fine
- DFS_NEAR_LOOP : One problem was found in of of the son
- DFS_LOOP_INSIDE : This node is part of a loop

Parameters **root** – Root of the dependency tree

Returns None

get_accessibility_packs()
Get accessibility packs of the graph: in one pack element are related in a way. Between packs, there is no relation at all. TODO: Make it work for directional graph too Because for now, edge must be father->son AND son->father

Returns packs of nodes

Return type list

loop_check()
Check if we have a loop in the graph

Returns Nodes in loop

Return type list

2.23 alignak.load module

2.24 alignak.log module

This module provides logging facilities for Alignak: - a main root logger for the running daemon - a monitoring log logger

It defines a colored stream handler class to allow using colored log. Using this class is as follows: alignak.log.ColorStreamHandler

It defines a CollectorHandler class that is used to easily capture the log events for the unit tests.

It also defines a UTC time formatter usable as alignak.log.UTCFormatter

The setup_logger function initializes the daemon logger with the JSON provided configuration file.

The make_monitoring_log function emits a log to the monitoring log logger and returns a brok for the Alignak broker.

class alignak.log.**CollectorHandler**
Bases: logging.Handler

This logging handler collects all the emitted logs in an inner list.

Note: This s only used for unit tests purpose

emit(*record*)
Do whatever it takes to actually log the specified logging record.

This version is intended to be implemented by subclasses and so raises a NotImplementedError.

class alignak.log.**ColorStreamHandler**(*stream=None*)
Bases: logging.StreamHandler

This logging handler provides colored logs when logs are emitted to a tty.

emit(*record*)
Emit a record.

If a formatter is specified, it is used to format the record. The record is then written to the stream with a trailing newline. If exception information is present, it is formatted using traceback.print_exception and appended to the stream. If the stream has an 'encoding' attribute, it is used to determine how to do the output to the stream.

class alignak.log.**UTCFormatter**(*fmt=None*, *datefmt=None*)
Bases: logging.Formatter

This logging formatter converts the log date/time to UTC

converter()

gmtime([seconds]) -> (tm_year, tm_mon, tm_mday, tm_hour, tm_min, tm_sec, tm_wday, tm_yday, tm_isdst)

Convert seconds since the Epoch to a time tuple expressing UTC (a.k.a. GMT). When 'seconds' is not passed in, convert the current time instead.

alignak.log.**get_log_level**()
Get the Alignak logger log level. This is used when getting the daemon log level thanks to the WS interface

Returns n/a

alignak.log.**make_monitoring_log**(*level*, *message*, *timestamp=None*, *to_logger=False*)
Function used to build the monitoring log.

Emit a log message with the provided level to the monitoring log logger. Build a Brok typed as monitoring_log with the provided message

When to_logger is True, the information is sent to the python logger, else a monitoring_log Brok is returned. The Brok is managed by the daemons to build an Event that will br logged by the Arbiter when it collects all the events.

TODO: replace with dedicated brok for each event to log - really useful?

Parameters

- **level** (*str*) – log level as defined in logging
- **message** (*str*) – message to send to the monitoring log logger
- **to_logger** (*bool*) – when set, send to the logger, else raise a brok
- **timestamp** – if set, force the log event timestamp

Returns a monitoring_log Brok

Return type *alignak.brok.Brok*

alignak.log.**set_log_console**(*log_level=20*)

Set the Alignak daemons logger have a console log handler.

This is only used for the arbiter verify mode to add a console log handler.

Parameters **log_level** – log level

Returns n/a

alignak.log.**set_log_level**(*log_level=20*, *handlers=None*)

Set the Alignak logger log level. This is mainly used for the arbiter verify code to set the log level at INFO level whatever the configured log level is set.

This is also used when changing the daemon log level thanks to the WS interface

If an handlers name list is provided, all the handlers which name is in this list are concerned else only the *daemons* handler log level is changed.

Parameters

- **handlers** – list of concerned handlers
- **log_level** – log level

Type list

Returns n/a

alignak.log.**setup_logger**(*logger_configuration_file*, *log_dir=None*, *process_name=''*, *log_file=''*)

Configure the provided logger - get and update the content of the Json configuration file - configure the logger with this file

If a log_dir and process_name are provided, the format and filename in the configuration file are updated with the provided values if they contain the patterns %(logdir)s and %(daemon)s

If no log_dir and process_name are provided, this function will truncate the log file defined in the configuration file.

If a log file name is provided, it will override the default defined log file name.

At first, this function checks if the logger is still existing and initialized to update the handlers and formatters. This mainly happens during the unit tests.

Parameters

- **logger_configuration_file** – Python Json logger configuration file
- **log_dir** – default log directory to update the defined logging handlers
- **process_name** – process name to update the defined logging formatters
- **log_file** – log file name to update the defined log file

Rtype logger_configuration_file str

Rtype log_dir str

Rtype process_name str

Rtype log_file str

Returns None

2.25 alignak.macroresolver module

This class resolve Macro in commands by looking at the macros list in Class of elements. It gives a property that may be a callable or not.

It not callable, it's a simple property and we replace the macro with the property value.

If callable, it's a method that is called to get the value. For example, to get the number of service in a host, you call a method to get the len(host.services)

class alignak.macroresolver.**MacroResolver**

Bases: *alignak.borg.Borg*

MacroResolver class is used to resolve macros (in command call). See above for details

get_env_macros(*data*)

Get all environment macros from data For each object in data

```
* Fetch all macros in object.__class__.macros
* Fetch all customs macros in o.custom
```

Parameters **data** – data to get macro

Returns dict with macro name as key and macro value as value

Return type dict

init(*conf*)

Initialize MacroResolver instance with conf. Must be called at least once.

Parameters **conf** (*alignak.objects.Config*) – configuration to load

Returns None

macros = {'DATE': '_get_date', 'EVENTSTARTTIME': '_get_events_start_time', 'LONGDATETI

my_type = 'macroresolver'

output_macros = ['HOSTOUTPUT', 'HOSTPERFDATA', 'HOSTACKAUTHOR', 'HOSTACKCOMMENT', 'SER

resolve_command(*com*, *data*, *macromodulations*, *timeperiods*)

Resolve command macros with data

Parameters

- **com** (*object*) – check / event handler or command call object
- **data** – objects list, used to search for a specific macro (custom or object related)
- **macromodulations** (*dict*) – the available macro modulations
- **timeperiods** (*dict*) – the available timeperiods

Returns command line with '$MACRO$' replaced with values

Return type str

resolve_simple_macros_in_string(*c_line*, *data*, *macromodulations*, *timeperiods*, *args=None*)

Replace macro in the command line with the real value

Parameters

- **c_line** (*str*) – command line to modify
- **data** – objects list, use to look for a specific macro
- **macromodulations** (*dict*) – the available macro modulations
- **timeperiods** (*dict*) – the available timeperiods
- **args** – args given to the command line, used to get "ARGN" macros.

Returns command line with '$MACRO$' replaced with values

Return type str

2.26 alignak.message module

This module provides Message class. Used for communication between daemon process (with queues)

class alignak.message.**Message**(*_type*, *data=None*, *source=None*)

Bases: object

This is a simple message class for communications between actionners and workers

get_data()

Getter of _data attribute

Returns Message data

Return type str

get_source()

Getter of _source attribute

Returns Message from (actionner/worker name)

Return type str

get_type()

Getter of _type attribute

Returns Message type

Return type str

my_type = 'message'

2.27 alignak.modulesmanager module

This module provides ModulesManager class. Used to load modules in Alignak

class alignak.modulesmanager.**ModulesManager**(*daemon*)

Bases: object

This class is used to manage modules and call callback

check_alive_instances()
Check alive instances. If not, log error and try to restart it

Returns None

clear_instances(*instances=None*)
Request to "remove" the given instances list or all if not provided

Parameters **instances** – instances to remove (all instances are removed if None)

Returns None

get_external_instances(*phase=None*)
Get a list of external instances (in a specific phase)

If phase is None, return all external instances whtever the phase

Parameters **phase** – phase to filter (never used)

Returns external instances list

Return type list

get_instances()
Create, init and then returns the list of module instances that the caller needs.

This method is called once the Python modules are loaded to initialize the modules.

If an instance can't be created or initialized then only log is doneand that instance is skipped. The previous modules instance(s), if any, are all cleaned.

Returns module instances list

Return type list

get_internal_instances(*phase=None*)
Get a list of internal instances (in a specific phase)

If phase is None, return all internal instances whtever the phase

Parameters **phase** – phase to filter (never used)

Returns internal instances list

Return type list

load(*modules*)
Load Python modules and check their usability

Parameters **modules** – list of the modules that must be loaded

Returns

load_and_init(*modules*)
Import, instantiate & "init" the modules we manage

Parameters **modules** – list of the managed modules

Returns True if no errors

remove_instance(*instance*)
Request to cleanly remove the given instance. If instance is external also shutdown it cleanly

Parameters **instance** (*object*) – instance to remove

Returns None

set_daemon_name(*daemon_name*)
Set the daemon name of the daemon which this manager is attached to and propagate this daemon name to our managed modules

Parameters **daemon_name** –

Returns

set_to_restart(*instance*)
Put an instance to the restart queue

Parameters **instance** (*object*) – instance to restart

Returns None

start_external_instances(*late_start=False*)
Launch external instances that are load correctly

Parameters **late_start** (*bool*) – If late_start, don't look for last_init_try

Returns None

stop_all()
Stop all module instances

Returns None

try_instance_init(*instance*, *late_start=False*)
Try to "initialize" the given module instance.

Parameters

- **instance** (*object*) – instance to init
- **late_start** (*bool*) – If late_start, don't look for last_init_try

Returns True on successful init. False if instance init method raised any Exception.

Return type bool

try_to_restart_deads()
Try to reinit and restart dead instances

Returns None

2.28 alignak.monitor module

This module group the Alignak self monitoring features.

Currently, it contains an Alignak Web Service client used to report Alignak status to an external monitoring application

class alignak.monitor.**MonitorConnection**(*endpoint='http://127.0.0.1:7773/ws'*)
Bases: object

Base class for Alignak Web Services client connection

static decode(*response*)
Decodes and returns the response as JSON (dict) or raise BackendException :param response: requests.response object :return: dict

get(*endpoint*, *params=None*)
Get items or item in alignak backend

If an error occurs, a BackendException is raised.

This method builds a response as a dictionary that always contains: _items and _status:

```
{
    u'_items': [
        ...
    ],
    u'_status': u'OK'
}
```

Parameters

- **endpoint** (*str*) – endpoint (API URL) relative from root endpoint
- **params** (*dict*) – parameters for the backend API

Returns dictionary as specified upper

Return type dict

get_response(*method*, *endpoint*, *headers=None*, *json=None*, *params=None*, *data=None*)
Returns the response from the requested endpoint with the requested method :param method: str. one of the methods accepted by Requests ('POST', 'GET', ...) :param endpoint: str. the relative endpoint to access :param params: (optional) Dictionary or bytes to be sent in the query string for the `Request`. :param data: (optional) Dictionary, bytes, or file-like object to send in the body of the `Request`. :param json: (optional) json to send in the body of the `Request`. :param headers: (optional) Dictionary of HTTP Headers to send with the `Request`. :return: Requests.response

get_token()
Get the stored backend token

get_url(*endpoint*)
Returns the formated full URL endpoint :param endpoint: str. the relative endpoint to access :return: str

login(*username*, *password*)
Log into the WS interface and get the authentication token

if login is: - accepted, returns True - refused, returns False

In case of any error, raises a BackendException

Parameters

- **username** (*str*) – login name
- **password** (*str*) – password
- **generate** (*str*) – Can have these values: enabled | force | disabled

Returns return True if authentication is successfull, otherwise False

Return type bool

logout()
Logout from the backend

Returns return True if logout is successfull, otherwise False

Return type bool

patch(*endpoint*, *data*)
Method to update an item

The headers must include an If-Match containing the object _etag. headers = {'If-Match': contact_etag}

The data dictionary contain the fields that must be modified.

If the patching fails because the _etag object do not match with the provided one, a BackendException is raised with code = 412.

If inception is True, this method makes e new get request on the endpoint to refresh the _etag and then a new patch is called.

If an HTTP 412 error occurs, a BackendException is raised. This exception is: - code: 412 - message: response content - response: backend response

All other HTTP error raises a BackendException. If some _issues are provided by the backend, this exception is: - code: HTTP error code - message: response content - response: JSON encoded backend response (including '_issues' dictionary ...)

If no _issues are provided and an _error is signaled by the backend, this exception is: - code: backend error code - message: backend error message - response: JSON encoded backend response

Parameters

- **endpoint** (*str*) – endpoint (API URL)
- **data** (*dict*) – properties of item to update
- **headers** (*dict*) – headers (example: Content-Type). 'If-Match' required
- **inception** (*bool*) – if True tries to get the last _etag

Returns dictionary containing patch response from the backend

Return type dict

post(*endpoint*, *data*, *files=None*, *headers=None*)
Create a new item

Parameters

- **endpoint** (*str*) – endpoint (API URL)
- **data** (*dict*) – properties of item to create
- **files** (*None*) – Not used. To be implemented
- **headers** (*dict*) – headers (example: Content-Type)

Returns response (creation information)

Return type dict

set_token(*token*)
Set token in authentification for next requests :param token: str. token to set in auth. If None, reinit auth

token
Get the stored backend token

2.29 alignak.notification module

This module provides Notification class. Used to define monitoring notifications (email, contacts..)

class alignak.notification.**Notification**(*params=None*, *parsing=False*)
Bases: *alignak.action.Action*

Notification class, inherits from action class. Used to notify contacts and execute notification command defined in configuration

ack_author

ack_data

already_start_escalations

author

author_alias

author_comment

author_name

check_time

command

command_call

contact

contact_name

creation_time

enable_environment_macros

end_time

env

escalated

execution_time

exit_status

fill_data_brok_from(*data*, *brok_type*)
Fill data with info of item by looking at brok_type in props of properties or running_properties

Parameters

- **data** – data to fill
- **brok_type** – type of brok

Returns brok with wanted data

Return type *alignak.brok.Brok*

get_initial_status_brok()
Get a initial status brok

Returns brok with wanted data

Return type *alignak.brok.Brok*

get_return_from(*notif*)
Setter of exit_status and execution_time attributes

Parameters **notif** (alignak.notification.Notification) – notification to get data from

Returns None

host_name

internal

is_a

is_administrative()
Check if this notification is "administrative"

Returns True in type not in ('PROBLEM', 'RECOVERY'), False otherwise

Return type bool

last_poll

long_output

macros = {'HOSTNOTIFICATIONID': 'uuid', 'HOSTNOTIFICATIONNUMBER': 'notif_nb', 'NOTIFIC

module_type

my_scheduler

my_type = 'notification'

my_worker

notif_nb

output

perf_data

properties = {'_in_timeout': <Property <class 'alignak.property.BoolProp'>, default:

reactionner_tag

reason_type

recipients

ref

ref_type

s_time

serialize()
This function serialize into a simple dict object. It is used when transferring data to other daemons over the network (http)

Here we directly return all attributes

Returns json representation of a Timeperiod

Return type dict

service_description

start_time

state

status

t_to_go

timeout

type

u_time

wait_time

2.30 alignak.property module

This module provides property classes. It is used during configuration parsing to ensure attribute type in objects. Each class implements a pythonize method that cast data into the wanted type.

class alignak.property.**UnusedProp**(*text=None*)
Bases: alignak.property.Property

A unused Property. These are typically used by Nagios but no longer useful/used by Alignak.

This is just to warn the user that the option he uses is no more used in Alignak.

class alignak.property.**BoolProp**(*default=<object object>*, *class_inherit=None*, *unmanaged=False*, *_help=''*, *no_slots=False*, *fill_brok=None*, *brok_transformation=None*, *retention=False*, *retention_preparation=None*, *retention_restoration=None*, *to_send=False*, *override=False*, *managed=True*, *split_on_comma=True*, *keep_empty=False*, *merging='uniq'*, *special=False*)
Bases: alignak.property.Property

A Boolean Property.

Boolean values are currently case insensitively defined as 0, false, no, off for False, and 1, true, yes, on for True).

pythonize(*val*)
Convert value into a boolean

Parameters **val** (*bool, int, str*) – value to convert

Returns boolean corresponding to value

{'1': True, 'yes': True, 'true': True, 'on': True, '0': False, 'no': False, 'false': False, 'off': False}

Return type bool

class alignak.property.**IntegerProp**(*default=<object object>*, *class_inherit=None*, *unmanaged=False*, *_help=''*, *no_slots=False*, *fill_brok=None*, *brok_transformation=None*, *retention=False*, *retention_preparation=None*, *retention_restoration=None*, *to_send=False*, *override=False*, *managed=True*, *split_on_comma=True*, *keep_empty=False*, *merging='uniq'*, *special=False*)
Bases: alignak.property.Property

Integer property

pythonize(*val*)
Convert value into an integer:

```
* If value is a list, try to take the last element
* Then call float(int(val))
```

Parameters **val** – value to convert

Returns integer corresponding to value

Return type int

class alignak.property.**FloatProp**(*default=<object object>*, *class_inherit=None*, *unmanaged=False*, *_help=''*, *no_slots=False*, *fill_brok=None*, *brok_transformation=None*, *retention=False*, *retention_preparation=None*, *retention_restoration=None*, *to_send=False*, *override=False*, *managed=True*, *split_on_comma=True*, *keep_empty=False*, *merging='uniq'*, *special=False*)

Bases: alignak.property.Property

Float property

pythonize(*val*)

Convert value into a float:

```
* If value is a list, try to take the last element
* Then call float(val)
```

Parameters **val** – value to convert

Returns float corresponding to value

Return type float

class alignak.property.**CharProp**(*default=<object object>*, *class_inherit=None*, *unmanaged=False*, *_help=''*, *no_slots=False*, *fill_brok=None*, *brok_transformation=None*, *retention=False*, *retention_preparation=None*, *retention_restoration=None*, *to_send=False*, *override=False*, *managed=True*, *split_on_comma=True*, *keep_empty=False*, *merging='uniq'*, *special=False*)

Bases: alignak.property.Property

One character string property

pythonize(*val*)

Convert value into a char

```
* If value is a list try, to take the last element
* Then take the first char of val (first elem)
```

Parameters **val** – value to convert

Returns char corresponding to value

Return type str

class alignak.property.**StringProp**(*default=<object object>*, *class_inherit=None*, *unmanaged=False*, *_help=''*, *no_slots=False*, *fill_brok=None*, *brok_transformation=None*, *retention=False*, *retention_preparation=None*, *retention_restoration=None*, *to_send=False*, *override=False*, *managed=True*, *split_on_comma=True*, *keep_empty=False*, *merging='uniq'*, *special=False*)

Bases: alignak.property.Property

String property

pythonize(*val*)

Convert value into a string:

```
* If value is a list, try to take the last element
```

Parameters **val** – value to convert

Returns str corresponding to value

Return type str

class alignak.property.**ListProp**(*default=<object object>*, *class_inherit=None*, *unmanaged=False*, *_help=''*, *no_slots=False*, *fill_brok=None*, *brok_transformation=None*, *retention=False*, *retention_preparation=None*, *retention_restoration=None*, *to_send=False*, *override=False*, *managed=True*, *split_on_comma=True*, *keep_empty=False*, *merging='uniq'*, *special=False*)

Bases: alignak.property.Property

List property

pythonize(*val*)

Convert value into a list:

```
* split value (or each element if value is a list) on coma char
* strip split values
```

Parameters **val** (*str*) – value to convert

Returns list corresponding to value

Return type list

class alignak.property.**DictProp**(*elts_prop=None*, **args*, ***kwargs*)

Bases: alignak.property.Property

Dict property

pythonize(*val*)

Convert value into a dict:

```
* If value is a list, try to take the last element
* split "key=value" string and convert to { key:value }
```

Parameters **val** – value to convert

Returns log level corresponding to value

Return type str

2.31 alignak.satellite module

This class is an interface for Reactionner and Poller daemons A Reactionner listens to a port for the configuration from the Arbiter The conf contains the schedulers where actionners will gather actions.

The Reactionner keeps on listening to the Arbiter

If Arbiter wants it to have a new conf, the satellite forgets the previous Schedulers (and actions into) and takes the new ones.

class alignak.satellite.**BaseSatellite**(*name*, ***kwargs*)
Bases: *alignak.daemon.Daemon*

Base Satellite class. Sub-classed by Alignak (scheduler), Broker and Satellite

clean_previous_run()
Clean variables from previous configuration, such as schedulers, broks and external commands

Returns None

do_loop_turn()
Satellite main loop - not implemented for a BaseSatellite

get_daemon_stats(*details=False*)
Increase the stats provided by the Daemon base class

Returns stats dictionary

Return type dict

get_events()
Get event list from satellite

Returns A copy of the events list

Return type list

get_external_commands()
Get the external commands

Returns External commands list

Return type list

get_managed_configurations()
Get the configurations managed by this satellite

The configurations managed by a satellite is a list of the configuration attached to the schedulers related to the satellites. A broker linked to several schedulers will return the list of the configuration parts of its scheduler links.

Returns a dict of scheduler links with instance_id as key and

hash, push_flavor and configuration identifier as values :rtype: dict

get_results_from_passive(*scheduler_instance_id*)
Get executed actions results from a passive satellite for a specific scheduler

Parameters **scheduler_instance_id** (*int*) – scheduler id

Returns Results list

Return type list

get_scheduler_from_hostname(*host_name*)
Get scheduler linked to the given host_name

Parameters **host_name** (*str*) – host_name we want the scheduler from

Returns scheduler with id corresponding to the mapping table

Return type dict

properties = {'active': <Property <class 'alignak.property.BoolProp'>, default: True

setup_new_conf()

Setup the new configuration received from Arbiter

This function is the generic treatment needed for every Alignak daemon when it receivss a new configuration from the Arbiter: - save the new configuration - dump the main configuration elements - get its own configuration (self_conf) - get its name and update the process title - set the timezone if needed - register its statistics manager - get and configure its arbiters and schedulers relation

Setting the self.new_conf as None is to indicate that the new configuration has been managed.

Note: it is important to protect the configuration management thanks to a lock!

Returns None

exception alignak.satellite.**NotWorkerMod**

Bases: exceptions.Exception

Class to tell that we are facing a non worker module but a standard one

class alignak.satellite.**Satellite**(*name*, ***kwargs*)

Bases: *alignak.satellite.BaseSatellite*

Satellite class. Sub-classed by Receiver, Reactionner and Poller

add(*elt*)

Generic function to add objects to the daemon internal lists. Manage Broks, External commands

Parameters **elt** (*alignak.AlignakObject*) – object to add

Returns None

add_actions(*actions_list*, *scheduler_instance_id*)

Add a list of actions to the satellite queues

Parameters

- **actions_list** (*list*) – Actions list to add
- **scheduler_instance_id** (SchedulerLink) – sheduler link to assign the actions to

Returns None

adjust_worker_number_by_load()

Try to create the minimum workers specified in the configuration

Returns None

assign_to_a_queue(*action*)

Take an action and put it to a worker actions queue

Parameters **action** (alignak.action.Action) – action to put

Returns None

check_and_del_zombie_workers()

Check if worker are fine and kill them if not. Dispatch the actions in the worker to another one

TODO: see if unit tests would allow to check this code?

Returns None

clean_previous_run()
Clean variables from previous configuration, such as schedulers, broks and external commands

Returns None

create_and_launch_worker(*module_name='fork'*)

Create and launch a new worker, and put it into self.workers It can be mortal or not

Parameters **module_name** (*str*) – the module name related to the worker default is "fork" for no module Indeed, it is actually the module 'python_name'

Returns None

do_actions = False

do_checks = False

do_get_new_actions()
Get new actions from schedulers Create a Message and put into the module queue REF: doc/alignak-action-queues.png (1)

Returns None

do_loop_turn()
Satellite main loop:

```
* Check and delete zombies actions / modules
* Get returns from queues
* Adjust worker number
* Get new actions
```

Returns None

do_post_daemon_init()
Do this satellite (poller or reactionner) post "daemonize" init

Returns None

do_stop()
Stop my workers and stop

Returns None

do_stop_workers()
Stop all workers

Returns None

get_broks()
Get brok list from satellite

Returns A copy of the broks list

Return type list

get_daemon_stats(*details=False*)
Increase the stats provided by the Daemon base class

Returns stats dictionary

Return type dict

get_new_actions()
Wrapper function for do_get_new_actions For stats purpose

Returns None

TODO: Use a decorator for timing this function

main()
Main satellite function. Do init and then mainloop

Returns None

manage_action_return(*action*)
Manage action return from Workers We just put them into the corresponding sched and we clean unused properties like my_scheduler

Parameters **action** (alignak.action.Action) – the action to manage

Returns None

my_type = ''

properties = {'active': <Property <class 'alignak.property.BoolProp'>, default: True

push_results()
Push the checks/actions results to our schedulers

Returns None

setup_new_conf()
Setup the new configuration received from Arbiter

This function calls the base satellite treatment and manages the configuration needed for a simple satellite daemon that executes some actions (eg. poller or reactionner): - configure the passive mode - configure the workers - configure the tags - configure the modules

Returns None

2.32 alignak.scheduler module

This module provides Scheduler class. It is used to schedule checks, create broks for monitoring event, handle downtime, problems / acknowledgment etc. The major part of monitoring "intelligence" is in this module.

class alignak.scheduler.**Scheduler**(*scheduler_daemon*)
Bases: object

Scheduler class. Mostly handle scheduling items (host service) to schedule checks raise alerts, manage downtimes, etc.

add(*elt*)
Generic function to add objects into the scheduler daemon internal lists:: Brok -> self.broks Check -> self.checks Notification -> self.actions EventHandler -> self.actions

For an ExternalCommand, tries to resolve the command

Parameters **elt** – element to add

Returns None

add_brok(*brok*, *broker_uuid=None*)
Add a brok into brokers list It can be for a specific one, all brokers or none (startup)

Parameters

- **brok** (alignak.brok.Brok) – brok to add
- **broker_uuid** (*str*) – broker uuid for the brok

Returns None

add_check(*check*)
Add a check into the scheduler checks list

Parameters **check** (alignak.check.Check) – check to add

Returns None

add_event_handler(*action*)
Add a event handler into actions list

Parameters **action** (alignak.eventhandler.EventHandler) – event handler to add

Returns None

add_external_command(*ext_cmd*)
Resolve external command

Parameters **ext_cmd** (alignak.external_command.ExternalCommand) – external command to run

Returns None

add_notification(*notification*)
Add a notification into actions list

Parameters **notification** (alignak.notification.Notification) – notification to add

Returns None

after_run()
After the scheduling process

all_my_hosts_and_services()
Create an iterator for all my known hosts and services

Returns None

before_run()
Initialize the scheduling process

check_for_expire_acknowledge()
Iter over host and service and check if any acknowledgement has expired

Returns None

check_freshness()
Iter over all hosts and services to check freshness if check_freshness enabled and passive_checks_enabled are set

For the host items, the list of hosts to check contains hosts that: - have freshness check enabled - are not yet freshness expired - are only passively checked

For the service items, the list of services to check contains services that: - do not depend upon an host that is freshness expired - have freshness check enabled - are not yet freshness expired - are only passively checked

Returns None

check_orphaned()
Check for orphaned checks/actions:

```
* status == 'in_poller' and t_to_go < now - time_to_orphanage (300 by␣
↪default)
```

if so raise a warning log.

Returns None

clean_caches()
Clean timperiods caches

Returns None

clean_queues()
Reduces internal list size to max allowed

- checks and broks : 5 * length of hosts + services
- actions : 5 * length of hosts + services + contacts

Returns None

consume_results()
Handle results waiting in waiting_results list. Check ref will call consume result and update their status

Returns None

delete_zombie_actions()
Remove actions that have a zombie status (usually timeouts)

Returns None

delete_zombie_checks()
Remove checks that have a zombie status (usually timeouts)

Returns None

dump_config()
Dump scheduler configuration into a temporary file

The dumped content is JSON formatted

Returns None

dump_objects()
Dump scheduler objects into a dump (temp) file

Returns None

fill_initial_broks(*broker_name*)
Create initial broks for a specific broker

Parameters **broker_name** (*str*) – broker name

Returns number of created broks

find_item_by_id(*object_id*)
Get item based on its id or uuid

Parameters **object_id** (*int | str*) –

Returns

Return type alignak.objects.item.Item | None

get_and_register_check_result_brok(*item*)
Get a check result brok for item and add it

Parameters **item** (alignak.objects.schedulingitem.SchedulingItem) – item to get brok from

Returns None

get_and_register_status_brok(*item*)
Get a update status brok for item and add it

Parameters **item** (alignak.objects.item.Item) – item to get brok from

Returns None

get_checks_status_counts(*checks=None*)
Compute the counts of the different checks status and return it as a defaultdict(int) with the keys being the different status and the values being the count of the checks in that status.

Checks None or the checks you want to count their statuses. If None then self.checks is used.

Parameters **checks** (*None | dict*) – NoneType | dict

Returns

Return type defaultdict(int)

get_latency_average_percentile()
Get a overview of the latencies with just a 95 percentile + min/max values

Returns None

get_new_actions()
Call 'get_new_actions' hook point Iter over all hosts and services to add new actions in internal lists

Returns None

get_new_broks()
Iter over all hosts and services to add new broks in internal lists

Returns None

get_objects_from_from_queues()
Same behavior than Daemon.get_objects_from_from_queues().

Returns

Return type

get_program_status_brok(*brok_type='program_status'*)
Create a program status brok

Initially builds the running properties and then, if initial status brok, get the properties from the Config class where an entry exist for the brok 'full_status'

Returns Brok with program status data

Return type *alignak.brok.Brok*

get_results_from_passive_satellites()
Get actions/checks results from passive poller/reactionners

Returns None

get_retention_data()
Get all hosts and services data to be sent to the retention storage.

This function only prepares the data because a module is in charge of making the data survive to the scheduler restart.

todo: Alignak scheduler creates two separate dictionaries: hosts and services It would be better to merge the services into the host dictionary!

Returns dict containing host and service data

Return type dict

get_scheduler_stats(*details=False*)
Get the scheduler statistics

Returns A dict with the following structure

```
{ 'modules': [
               {'internal': {'name': "MYMODULE1", 'state': 'ok'},
               {'external': {'name': "MYMODULE2", 'state': 'stopped'},
             ]
  'latency':  {'avg': lat_avg, 'min': lat_min, 'max': lat_max}
  'hosts': len(self.hosts),
  'services': len(self.services),
  'commands': [{'cmd': c, 'u_time': u_time, 's_time': s_time}, ...] (10␣
↪first)
  'livesynthesis': {...}
}
```

Return type dict

get_to_run_checks(*do_checks=False*, *do_actions=False*, *poller_tags=None*, *reactionner_tags=None*, *worker_name='none'*, *module_types=None*)
Get actions/checks for reactionner/poller

Called by the poller to get checks (do_checks=True) and by the reactionner (do_actions=True) to get actions

Parameters

- **do_checks** (*bool*) – do we get checks ?
- **do_actions** (*bool*) – do we get actions ?
- **poller_tags** (*list*) – poller tags to filter
- **reactionner_tags** (*list*) – reactionner tags to filter
- **worker_name** (*str*) – worker name to fill check/action (to remember it)
- **module_types** (*list*) – module type to filter

Returns Check/Action list with poller/reactionner tags matching and module type matching

Return type list

hook_point(*hook_name*)
Generic function to call modules methods if such method is avalaible

Parameters **hook_name** (*str*) – function name to call

:return:None

initial_program_status()
Create and add a program_status brok

Returns None

load_conf(*instance_id*, *instance_name*, *conf*)
Load configuration received from Arbiter and pushed by our Scheduler daemon

Parameters

- **instance_name** (*str*) – scheduler instance name
- **instance_id** (*str*) – scheduler instance id
- **conf** (alignak.objects.config.Config) – configuration to load

Returns None

log_initial_states()
Raise hosts and services initial status logs

First, raise hosts status and then services. This to allow the events log to be a little sorted.

Returns None

manage_internal_checks()
Run internal checks

Returns None

manage_results(*action*)
Get result from pollers/reactionners (actives ones)

Parameters **action** – check / action / event handler to handle

Returns None

name
Get the scheduler name

Indeed, we return our suffixed daemon name

Returns

Return type

push_actions_to_passive_satellites()
Send actions/checks to passive poller/reactionners

Returns None

reset()
Reset scheduler:

```
* Remove waiting results
* Clear checks and actions lists
```

Returns None

reset_topology_change_flag()
Set topology_change attribute to False in all hosts and services

Returns None

restore_retention_data(*data*)
Restore retention data

Data coming from retention will override data coming from configuration It is kinda confusing when you modify an attribute (external command) and it get saved by retention

Parameters **data** (*dict*) – data from retention

Returns None

restore_retention_data_item(*data*, *item*)
Restore data in item

Parameters

- **data** (*dict*) – retention data of the item
- **item** (*alignak.objects.host.Host | alignak.objects.service.Service*) – host or service item

Returns None

retention_load(*forced=False*)
Call hook point 'load_retention'. Retention modules will read retention (from file, db etc)

Parameters **forced** (*bool*) – is load forced?

Returns None

run()
Main scheduler function:

```
* Load retention
* Call 'pre_scheduler_mod_start' hook point
* Start modules
* Schedule first checks
* Init connection with pollers/reactionners
* Run main loop
```

- Do recurrent works
- Push/Get actions to passive satellites
- Update stats
- Call 'scheduler_tick' hook point
- Save retention (on quit)

Returns None

run_external_commands(*cmds*)
Run external commands Arbiter/Receiver sent

Parameters **cmds** (*list*) – commands to run

Returns None

scatter_master_notifications()
Generate children notifications from a master notification Also update notification number

Master notification are raised when a notification must be sent out. They are not launched by reactionners (only children are) but they are used to build the children notifications.

From one master notification, several children notifications may be built, indeed one per each contact...

Returns None

schedule(*elements=None*)
Iterate over all hosts and services and call schedule method (schedule next check)

If elements is None all our hosts and services are scheduled for a check.

Parameters **elements** (*None | list*) – None or list of host / services to schedule

Returns None

send_broks_to_modules()
Put broks into module queues Only broks without sent_to_externals to True are sent Only modules that ask for broks will get some

Returns None

start_scheduling()
Set must_schedule attribute to True - enable the scheduling loop

Returns None

stop_scheduling()
Set must_schedule attribute to False - disable the scheduling loop

Returns None

update_business_values()
Iter over host and service and update business_impact

Returns None

update_downtimes_and_comments()
Iter over all hosts and services:

TODO: add some unit tests for the maintenance period feature.

- Update downtime status (start / stop) regarding maintenance period
- Register new comments in comments list

Returns None

update_program_status()
Create and add a update_program_status brok

Returns None

update_recurrent_works_tick(*conf*)
Modify the tick value for the scheduler recurrent work

A tick is an amount of loop of the scheduler before executing the recurrent work

The provided configuration may contain some tick-function_name keys that contain a tick value to be updated. Those parameters are defined in the alignak environment file.

Indeed this function is called with the Scheduler daemon object. Note that the `conf` parameter may also be a dictionary.

Parameters **conf** (`alignak.daemons.schedulerdaemon.Alignak`) – the daemon link configuration to search in

Returns None

update_retention()
Call hook point 'save_retention'. Retention modules will write back retention (to file, db etc)

Parameters **forced** (*bool*) – is update forced?

Returns None

2.33 alignak.stats module

This module allows to export Alignak internal metrics to a statsd server.

The *register* function allows an Alignak daemon to register some metrics and the expected behavior (sends to StatsD server and/or build an internal brok).

The *connect* function allows an Alignak daemon to connectto a Carbon/Graphite server.

As such it:

- registers the StatsD or Carbon) connexion parameters
- tries to establish a connection if the StatsD sending is enabled
- creates an inner dictionary for the registered metrics

If some environment variables exist the metrics will be logged to a file in append mode:

'ALIGNAK_STATS_FILE' the file name

'ALIGNAK_STATS_FILE_LINE_FMT' defaults to [#date#] #counter# #value# #uom#

'

'ALIGNAK_STATS_FILE_DATE_FMT' defaults to '%Y-%m-%d %H:%M:%S' date is UTC if configured as an empty string, the date will be output as a UTC timestamp

Every time a metric is updated thanks to the provided functions, the inner dictionary is updated according to keep the last value, the minimum/maximum values, to update an internal count of each update and to sum the collected values. **Todo**: Interest of this feature is to be proven ;)

The *timer* function sends a timer value to the StatsD registered server and creates an internal brok.

The *counter* function sends a counter value to the StatsD registered server and creates an internal brok.

The *gauge* function sends a gauge value to the StatsD registered server and creates an internal brok.

class alignak.stats.**Stats**

Bases: object

Stats class to export data into a statsd (or carbon/Graphite) format

This class allows to send metrics to a StatsD server using UDP datagrams. Same behavior as:

```
echo "foo:1|c" | nc -u -w0 127.0.0.1 8125
```

With the Graphite option, this class stores the metrics in an inner list and flushes the metrics to a Graphite instance when the flush method is called.

connect(*name*, *_type*, *host='localhost'*, *port=2004*, *prefix='alignak'*, *enabled=False*, *broks_enabled=False*)

Init instance with real values for a graphite/carbon connection

Parameters

- **name** (*str*) – daemon name
- **_type** – daemon type
- **host** (*str*) – host to post data
- **port** (*int*) – port to post data
- **prefix** (*str*) – prefix to add to metric
- **enabled** (*bool*) – bool to enable statsd

- **broks_enabled** (*bool*) – bool to enable broks sending

Returns None

counter(*key*, *value*, *timestamp=None*)
Set a counter value

If the inner key does not exist is is created

Parameters

- **key** (*str*) – counter to update
- **value** (*float*) – counter value

Returns An alignak_stat brok if broks are enabled else None

flush(*log=False*)
Send inner stored metrics to the defined Graphite

Returns False if the sending failed with a warning log if log parameter is set

Returns bool

gauge(*key*, *value*, *timestamp=None*)
Set a gauge value

If the inner key does not exist is is created

Parameters

- **key** (*str*) – gauge to update
- **value** (*float*) – counter value

Returns An alignak_stat brok if broks are enabled else None

load_statsd()
Create socket connection to statsd host

Note that because of the UDP protocol used by StatsD, if no server is listening the socket connection will be accepted anyway :)

Returns True if socket got created else False and an exception log is raised

metrics_count
Number of internal stored metrics :return:

register(*name*, *_type*, *statsd_host='localhost'*, *statsd_port=8125*, *statsd_prefix='alignak'*, *statsd_enabled=False*, *broks_enabled=False*)
Init instance with real values

Parameters

- **name** (*str*) – daemon name
- **_type** – daemon type
- **statsd_host** (*str*) – host to post data
- **statsd_port** (*int*) – port to post data
- **statsd_prefix** (*str*) – prefix to add to metric
- **statsd_enabled** (*bool*) – bool to enable statsd
- **broks_enabled** (*bool*) – bool to enable broks sending

Returns None

send_to_graphite(*metric*, *value*, *timestamp=None*)
Inner store a new metric and flush to Graphite if the flush threshold is reached.

If no timestamp is provided, get the current time for the metric timestam.

Parameters

- **metric** (*str*) – metric name in dotted format
- **value** (*float*) –
- **timestamp** (*int*) – metric timestamp

timer(*key*, *value*, *timestamp=None*)
Set a timer value

If the inner key does not exist is is created

Parameters

- **key** (*str*) – timer to update
- **value** (*float*) – timer value (in seconds)
- **timestamp** (*int*) – metric timestamp

Returns An alignak_stat brok if broks are enabled else None

2.34 alignak.util module

This module provide a lot of utility functions. You can find functions for time management, type management (pythonization), macros solving, sorting, parsing, file handling, filters.

exception alignak.util.**KeyValueSyntaxError**
Bases: exceptions.ValueError

Syntax error on a duplicate_foreach value

alignak.util.**average_percentile**(*values*)
Get the average, min percentile (5%) and max percentile (95%) of a list of values.

Parameters **values** (*list*) – list of value to compute

Returns tuple containing average, min and max value

Return type tuple

alignak.util.**brok_last_time**(*ref*, *val*)
Convert float to int

Parameters

- **ref** – Not used
- **val** (*float*) – value to convert

Returns int(val)

Return type int

alignak.util.**dict_to_serialized_dict**(*ref*, *the_dict*)
Serialize the list of elements to a dictionary

Used for the retention store

Parameters

- **ref** – Not used
- **the_dict** (*dict*) – dictionary to convert

Returns dict of serialized

Return type dict

alignak.util.**expand_ranges**(*value*)

Parameters **value** (*str*) – The value to be "expanded".

Returns A generator to yield the different resulting values from expanding the eventual ranges present in the input value.

```
>>> tuple(expand_ranges("Item [1-3] - Bla"))
('Item 1 - Bla', 'Item 2 - Bla', 'Item 3 - Bla')
>>> tuple(expand_ranges("X[1-10/2]Y"))
('X1Y', 'X3Y', 'X5Y', 'X7Y', 'X9Y')
>>> tuple(expand_ranges("[1-6/2] [1-3]"))
('1 1', '1 2', '1 3', '3 1', '3 2', '3 3', '5 1', '5 2', '5 3')
```

alignak.util.**filter_any**(*ref*)
Filter for host Filter nothing

Parameters **name** (*str*) – name to filter

Returns Filter

Return type bool

alignak.util.**filter_host_by_bp_rule_label**(*label*)
Filter for host Filter on label

Parameters **label** (*str*) – label to filter

Returns Filter

Return type bool

alignak.util.**filter_host_by_group**(*group*)
Filter for host Filter on group

Parameters **group** (*str*) – group name to filter

Returns Filter

Return type bool

alignak.util.**filter_host_by_name**(*name*)
Filter for host Filter on name

Parameters **name** (*str*) – name to filter

Returns Filter

Return type bool

alignak.util.**filter_host_by_regex**(*regex*)
Filter for host Filter on regex

Parameters **regex** (*str*) – regex to filter

Returns Filter

Return type bool

alignak.util.**filter_host_by_tag**(*tpl*)
Filter for host Filter on tag

Parameters **tpl** (*str*) – tag to filter

Returns Filter

Return type bool

alignak.util.**filter_none**(*ref*)
Filter for host Filter all

Returns Filter

Return type bool

alignak.util.**filter_service_by_bp_rule_label**(*label*)
Filter for service Filter on label

Parameters **label** (*str*) – label to filter

Returns Filter

Return type bool

alignak.util.**filter_service_by_host_bp_rule_label**(*label*)
Filter for service Filter on label

Parameters **label** (*str*) – label to filter

Returns Filter

Return type bool

alignak.util.**filter_service_by_host_name**(*host_name*)
Filter for service Filter on host_name

Parameters **host_name** (*str*) – host_name to filter

Returns Filter

Return type bool

alignak.util.**filter_service_by_host_tag_name**(*tpl*)
Filter for service Filter on tag

Parameters **tpl** (*str*) – tag to filter

Returns Filter

Return type bool

alignak.util.**filter_service_by_hostgroup_name**(*group*)
Filter for service Filter on hostgroup

Parameters **group** (*str*) – hostgroup to filter

Returns Filter

Return type bool

alignak.util.**filter_service_by_name**(*name*)
Filter for service Filter on name

Parameters **name** (*str*) – name to filter

Returns Filter

Return type bool

alignak.util.**filter_service_by_regex_host_name**(*regex*)
Filter for service Filter on regex host_name

Parameters **regex** (*str*) – regex to filter

Returns Filter

Return type bool

alignak.util.**filter_service_by_regex_name**(*regex*)
Filter for service Filter on regex

Parameters **regex** (*str*) – regex to filter

Returns Filter

Return type bool

alignak.util.**filter_service_by_servicegroup_name**(*group*)
Filter for service Filter on group

Parameters **group** (*str*) – group to filter

Returns Filter

Return type bool

alignak.util.**format_t_into_dhms_format**(*timestamp*)
Convert an amount of second into day, hour, min and sec

Parameters **timestamp** (*int*) – seconds

Returns 'Ad Bh Cm Ds'

Return type str

```
>>> format_t_into_dhms_format(456189)
'5d 6h 43m 9s'
```

```
>>> format_t_into_dhms_format(3600)
'0d 1h 0m 0s'
```

alignak.util.**from_bool_to_int**(*boolean*)
Convert a bool to a int representation

Parameters **boolean** (*bool*) – bool to convert

Returns if boolean 1 ,else 0

Return type int

alignak.util.**from_bool_to_string**(*boolean*)
Convert a bool to a string representation

Parameters **boolean** (*bool*) – bool to convert

Returns if boolean '1' ,else '0'

Return type str

alignak.util.**from_float_to_int**(*val*)
Convert float to int

Parameters **val** (*float*) – value to convert

Returns int(val)

Return type int

alignak.util.**from_list_to_set**(*ref*, *tab*)
Convert list to a set

Used for the retention restore

Parameters

- **ref** – Not used
- **tab** (*list*) – list to parse

Returns list of names

Return type list

alignak.util.**from_list_to_split**(*val*)
Convert list into a comma separated string

Parameters **val** – value to convert

Returns comma separated string

Return type str

alignak.util.**from_serialized**(*ref*, *the_data*)
Unserialize the element

Used for the retention store

Parameters

- **ref** – Not used
- **the_data** (*dict*) – dictionary to convert

Returns serialized data

Return type dict

alignak.util.**from_set_to_list**(*ref*, *tab*)
Convert set into a list

Used for the retention store

Parameters

- **ref** – Not used
- **tab** (*list*) – list to parse

Returns list of names

Return type list

alignak.util.**generate_key_value_sequences**(*entry*, *default_value*)
Parse a key value config entry (used in duplicate foreach)

If we have a key that look like [X-Y] we will expand it into Y-X+1 keys

Parameters

- **entry** (*str*) – The config line to be parsed.
- **default_value** (*str*) – The default value to be used when none is available.

Returns a generator yielding dicts with 'KEY' & 'VALUE' & 'VALUE1' keys, with eventual others 'VALUEx' (x 1 -> N) keys.

```
>>> rsp = list(generate_key_value_sequences("var$(/var)$,root $(/)$"))
>>> import pprint
>>> pprint.pprint(rsp)
[{'KEY': 'var', 'VALUE': '/var', 'VALUE1': '/var'},
 {'KEY': 'root', 'VALUE': '/', 'VALUE1': '/'}]
```

alignak.util.**get_obj_name_two_args_and_void**(*obj*, *value*)
Get value name (call get_name) if not a string

Parameters

- **obj** (*object*) – Not used
- **value** – value to name

Returns value name

Return type str

alignak.util.**is_complex_expr**(*expr*)
Check if expression in complex

Parameters **expr** (*str*) – expression to parse

Returns True if '(', ')', '&', 'l', '!' or '*' are in expr

Return type bool

alignak.util.**jsonify_r**(*obj*)
Convert an object into json (recursively on attribute)

Parameters **obj** (*object*) – obj to jsonify

Returns json representation of obj

Return type dict

alignak.util.**list_split**(*val*, *split_on_comma=True*)
Try to split each member of a list with comma separator. If we don't have to split just return val

Parameters

- **val** – value to split
- **split_on_comma** (*bool*) –

Returns list with members split on comma

Return type list

```
>>> list_split(['a,b,c'], False)
['a,b,c']
```

```
>>> list_split(['a,b,c'])
['a', 'b', 'c']
```

```
>>> list_split('')
[]
```

alignak.util.**list_to_serialized**(*ref*, *the_list*)
Serialize the list of elements

Used for the retention store

Parameters

- **ref** – Not used
- **the_list** (*dict*) – dictionary to convert

Returns dict of serialized

Return type dict

alignak.util.**master_then_spare**(*data*)

Return the provided satellites list sorted as:

- alive first,
- then spare
- then dead

satellites.

Parameters **data** (*list*) – the SatelliteLink list

Returns sorted list

Return type list

alignak.util.**merge_periods**(*data*)

Merge periods to have better continous periods. Like 350-450, 400-600 => 350-600

Parameters **data** (*list*) – list of periods

Returns better continous periods

Return type list

alignak.util.**parse_daemon_args**(*arbiter=False*)

Generic parsing function for daemons

All daemons: '-n', "–name": Set the name of the daemon to pick in the configuration files. This allows an arbiter to find its own configuration in the whole Alignak configuration Using this parameter is mandatory for all the daemons except for the arbiter (defaults to arbiter-master). If several arbiters are existing in the configuration this will allow to determine which one is the master/spare. The spare arbiter must be launched with this parameter!

'-e', '–environment': Alignak environment file - the most important and mandatory parameter to define the name of the alignak.ini configuration file

'-c', '–config': Daemon configuration file (ini file) - deprecated! '-d', '–daemon': Run as a daemon '-r', '–replace': Replace previous running daemon '-f', '–debugfile': File to dump debug logs.

These parameters allow to override the one defined in the Alignak configuration file: '-o', '–host': interface the daemon will listen to '-p', '–port': port the daemon will listen to

'-l', '–log_file': set the daemon log file name '-i', '–pid_file': set the daemon pid file name

Arbiter only: "-a", "–arbiter": Monitored configuration file(s), (multiple -a can be used, and they will be concatenated to make a global configuration file) - Note that this parameter is not necessary anymore "-V", "–verify-config": Verify configuration file(s) and exit

Parameters **arbiter** (*bool*) – Do we parse args for arbiter?

Returns args

alignak.util.**sort_by_number_values**(*x00*, *y00*)
Compare x00, y00 base on number of values

Parameters

- **x00** (*list*) – first elem to compare
- **y00** (*list*) – second elem to compare

Returns x00 > y00 (-1) if len(x00) > len(y00), x00 == y00 (0) if id equals, x00 < y00 (1) else

Return type int

alignak.util.**split_semicolon**(*line*, *maxsplit=None*)
Split a line on semicolons characters but not on the escaped semicolons

Parameters

- **line** (*str*) – line to split
- **maxsplit** (*None | int*) – maximal number of split (if None, no limit)

Returns split line

Return type list

```
>>> split_semicolon('a,b;c;;g')
['a,b', 'c', '', 'g']
```

```
>>> split_semicolon('a,b;c;;g', 2)
['a,b', 'c', ';g']
```

```
>>> split_semicolon(r'a,b;c\;;g', 2)
['a,b', 'c;', 'g']
```

alignak.util.**strip_and_uniq**(*tab*)
Strip every element of a list and keep a list of ordered unique values

Parameters **tab** (*list*) – list to strip

Returns stripped list with unique values

Return type list

alignak.util.**to_best_int_float**(*val*)
Get best type for value between int and float

Parameters **val** – value

Returns int(float(val)) if int(float(val)) == float(val), else float(val)

Return type int | float

```
>>> to_best_int_float("20.1")
20.1
```

```
>>> to_best_int_float("20.0")
20
```

```
>>> to_best_int_float("20")
20
```

alignak.util.**to_bool**(*val*)
Convert value to bool

Because bool('0') = true, so...

Parameters **val** – value to convert

Returns True if val == '1' or val == 'on' or val == 'true' or val == 'True', else False

Return type bool

alignak.util.**to_char**(*val*)
Get first character of val (or raise Exception)

Parameters **val** – value we get head

Returns val[0]

Return type str

alignak.util.**to_float**(*val*)
Convert val to float (or raise Exception)

Parameters **val** – value to convert

Returns float(val)

Return type float

alignak.util.**to_hostnames_list**(*ref*, *tab*)
Convert Host list into a list of host_name

Parameters

- **ref** – Not used
- **tab** (*list[alignak.objects.host.Host]*) – Host list

Returns host_name list

Return type list

alignak.util.**to_int**(*val*)
Convert val to int (or raise Exception)

Parameters **val** – value to convert

Returns int(float(val))

Return type int

alignak.util.**to_list_string_of_names**(*ref*, *tab*)
Convert list into a comma separated list of element name

Parameters

- **ref** – Not used
- **tab** (*list*) – list to parse

Returns comma separated string of names

Return type str

alignak.util.**to_name_if_possible**(*ref*, *value*)
Try to get value name (call get_name method)

Parameters

- **ref** – Not used
- **value** (*str*) – value to name

Returns name or ‘’

Return type str

alignak.util.**to_serialized**(*ref*, *the_data*)
Serialize the property

Used for the retention store

Parameters

- **ref** – Not used
- **the_data** (*dict*) – dictionary to convert

Returns serialized data

Return type dict

alignak.util.**to_split**(*val*, *split_on_comma=True*)
Try to split a string with comma separator. If val is already a list return it If we don’t have to split just return [val] If split gives only [‘’] empty it

Parameters

- **val** – value to split
- **split_on_comma** (*bool*) –

Returns split value on comma

Return type list

```
>>> to_split('a,b,c')
['a', 'b', 'c']
```

```
>>> to_split('a,b,c', False)
['a,b,c']
```

```
>>> to_split(['a,b,c'])
['a,b,c']
```

```
>>> to_split('')
[]
```

alignak.util.**to_svc_hst_distinct_lists**(*ref*, *tab*)
create a dict with 2 lists:

```
* services: all services of the tab
* hosts: all hosts of the tab
```

Parameters

- **ref** – Not used
- **tab** (*list*) – list of Host and Service

Returns dict with hosts and services names

Return type dict

alignak.util.**unique_value**(*val*)
Get last element of a value if it is a list else returns the value

Used in parsing, if we set several time a parameter we only take the last one

Parameters **val** – val to edit

Returns single value

Return type str

2.35 alignak.version module

This module provide Alignak current version

2.36 alignak.worker module

This module provide Worker class. It is used to spawn new processes in Poller and Reactionner

class alignak.worker.**Worker**(*module_name*, *actions_queue*, *returns_queue*, *processes_by_worker*, *timeout=300*, *max_plugins_output_length=8192*, *target=None*, *loaded_into='unknown'*)
Bases: object

This class is used for poller and reactionner to work. The worker is a process launch by theses process and read Message in a Queue (self.actions_queue) They launch the Check and then send the result in the Queue self.m (master) they can die if they do not do anything (param timeout)

check_for_system_time_change()
Check if our system time change. If so, change our

Returns 0 if the difference < 900, difference else

Return type int

do_work(*actions_queue*, *returns_queue*, *control_queue=None*)
Main function of the worker. * Get checks * Launch new checks * Manage finished checks

Parameters

- **actions_queue** (*Queue.Queue*) – Global Queue Master->Slave
- **returns_queue** (*Queue.Queue*) – queue managed by manager

Returns None

get_id()
Accessor to get the worker identifier

Returns the worker auto-generated identifier

Return type str

get_module()
Accessor to get the worker module name

Returns the worker module name

Return type str

get_new_checks(*queue*, *return_queue*)
Get new checks if less than nb_checks_max If no new checks got and no check in queue, sleep for 1 sec REF: doc/alignak-action-queues.png (3)

Returns None

get_pid()
Accessor to get the worker process PID

Returns the worker PID

Return type int

is_alive()
Wrapper for calling is_alive method of the process attribute

Returns A boolean indicating if the process is alive

Return type bool

join(*timeout=None*)
Wrapper for calling join method of the process attribute

Parameters **timeout** (*int*) – time to wait for the process to terminate

Returns None

launch_new_checks()
Launch checks that are in status REF: doc/alignak-action-queues.png (4)

Returns None

manage_finished_checks(*queue*)
Check the status of checks if done, return message finished :) REF: doc/alignak-action-queues.png (5)

Returns None

manage_signal(*sig*, *frame*)
Manage signals caught by the process but I do not do anything... our master daemon is managing our termination.

Parameters

- **sig** (*str*) – signal caught by daemon
- **frame** – current stack frame

Returns None

set_exit_handler()
Set the signal handler to manage_signal (defined in this class) Only set handlers for signal.SIGTERM, signal.SIGINT, signal.SIGUSR1, signal.SIGUSR2

Returns None

start()
Start the worker. Wrapper for calling start method of the process attribute

Returns None

terminate()
Wrapper for calling terminate method of the process attribute Also close queues (input and output) and terminate queues thread

Returns None

uuid = ''

work(*actions_queue*, *returns_queue*, *control_queue=None*)
Wrapper function for do_work in order to catch the exception to see the real work, look at do_work

Parameters

- **actions_queue** (*Queue.Queue*) – Global Queue Master->Slave
- **returns_queue** (*Queue.Queue*) – queue managed by manager

Returns None

2.37 Module contents

Init of Alignak, basically only import version and shinken_hook. This file has to be as small as possible in order to namespace to work.

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

Z