Alignak backend client Documentation

Release 1.4.4

Frederic Mohier

Sep 02, 2018

Contents

CHAPTER 1

Project version:

Main version: 1.4, release: 1.4.4

CHAPTER 2

Documentation content:

This documentation is built in two parts:

- the backend command line interface
- the backend client Python API

The first part introduces and provides some examples of a command line client that may be used to make simple operations with the Alignak backend.

The second part is a programmatic Python API to interact with the backend.

2.1 Backend client CLI

2.1.1 Alignak backend command line interface

`alignak_backend_cli` is an utility tool that can make simple operations with the Alignak backend. It allows getting, updating, creating data from/into the backend.

All the operations that this script permits are strongly related to the Alignak backend data model. Do not even expect dealing with hosts, services, commands, ... if you do not even have the lightest idea of what it is about;)

The Alignak backend data model is defined in the Alignak backend documentation and the best source of information is achieved with the Swagger interface of the backend. You run the backend and it provides its data model on a Web interface...

2.1.2 Usage

The `alignak_backend_cli` script receives some command line parameters to define its behavior:

Command line interface

alignak-backend-cli command line interface:

```
Usage:
    alignak-backend-cli [-h]
    alignak-backend-cli [-V]
    alignak-backend-cli [-v] [-q] [-c] [-l] [-m] [-e] [-i]
                        [-b=url] [-u=username] [-p=password]
                        [-d=data]
                        [-f=folder]
                        [-T=template] [-t=type] [<action>] [<item>]

Options:
    -h, --help                  Show this screen.
    -V, --version               Show application version.
    -v, --verbose               Run in verbose mode (more info to display)
    -q, --quiet                 Run in quiet mode (display nothing)
    -c, --check                 Check only (dry run), do not change the backend.
    -l, --list                  Get an items list
    -b, --backend url           Specify backend URL [default: http://127.0.0.1:5000]
    -u, --username=username     Backend login username [default: admin]
    -p, --password=password     Backend login password [default: admin]
    -d, --data=data             Data for the new item to create [default: none]
    -f, --folder=folder         Folder where to read/write data files [default: none]
    -i, --include-read-data     Do not use only the provided data, but append the one
                                read from he backend
    -t, --type=host             Type of the provided item [default: host]
    -e, --embedded              Do not embed linked objects
    -m, --model                 Get only the templates
    -T, --template=template     Template to use for the new item

Exit code:
    0 if required operation succeeded
    1 if backend access is denied (check provided username/password)
    2 if element operation failed (missing template,...)

    64 if command line parameters are not used correctly

Use cases:
    Display help message:
        alignak-backend-cli (-h | --help)

    Display current version:
        alignak-backend-cli -V
        alignak-backend-cli --version

    Specify backend parameters if they are different from the default
        alignak-backend-cli -b=http://127.0.0.1:5000 -u=admin -p=admin get host_name

Actions:
    'get' to get an item in the backend
    'list' (shortcut for 'get -l' to get the list of all items of a type
    'add' to add an(some) item(s) in the backend
    'update' to update an(some) item(s) in the backend
    'delete' to delete an item (or all items of a type) in the backend

Use cases to get data:
```

(continues on next page)

(continued from previous page)

```
    Get an items list from the backend:
        alignak-backend-cli get -l
        Try to get the list of all hosts and copy the JSON dump in a file named
        './alignak-object-list-hosts.json'

        alignak-backend-cli get -l -t user
        Try to get the list of all users and copy the JSON dump in a file named
        './alignak-object-list-users.json'

        alignak-backend-cli get -l -f /tmp -t user
        Try to get the list of all users and copy the JSON dump in a file named
        '/tmp/alignak-object-list-users.json'

        alignak-backend-cli list -t user
        Shortcut for 'alignak-backend-cli get -l -t user'

    Get the hosts templates list from the backend:
        alignak-backend-cli -l -m
        Try to get the list of all hosts templates and copy the JSON dump in a
        file named './alignak-object-list-hosts.json'

    Get an item from the backend:
        alignak-backend-cli get host_name
        Try to get the definition of an host named 'host_name' and copy the JSON dump
        in a file named './alignak-object-dump-host-host_name.json'

        alignak-backend-cli -t user get contact_name
        Try to get the definition of a user (contact) contact named 'contact_name' and
        copy the JSON dump in a file named './alignak-object-dump-contact-contact_
↪name.json'

    Get a service from the backend:
        alignak-backend-cli get -t service host_name/service_name
        Try to get the definition of the service service_name for an host named 'host_
↪name'
        and copy the JSON dump in a file named
        './alignak-object-dump-service-host_name_service_name.json'

Use cases to add data:
    Add an item to the backend (without templating):
        alignak-backend-cli new_host
        This will add an host named new_host

        alignak-backend-cli -t user new_contact
        This will add a user named new_contact

    Add an item to the backend (with some data):
        alignak-backend-cli --data="/tmp/input_host.json" add new_host
        This will add an host named new_host with the data that are read from the
        JSON file /tmp/input_host.json

        alignak-backend-cli -t user new_contact --data="stdin"
        This will add a user named new_contact with the JSON data read from the
        stdin. You can 'cat file > alignak-backend-cli -t user new_contact --data=
↪"stdin"'

    Add an item to the backend based on a template:
```

(continues on next page)

(continued from previous page)

```
        alignak-backend-cli -T host_template add new_host
        This will add an host named new_host with the data existing in the template
        host_template

    Add an item to the backend based on several templates:
        alignak-backend-cli -T "host_template,host_template2" add new_host
        This will add an host named new_host with the data existing in the templates
        host_template and host_template2

Use cases to update data:
    Update an item into the backend (with some data):
        alignak-backend-cli --data="./update_host.json" update test_host
        This will update an host named test_host with the data that are read from the
        JSON file ./update_host.json

Use cases to delete data:
    Delete an item from the backend:
        alignak-backend-cli delete test_host
        This will delete the host named test_host

    Delete all items from the backend:
        alignak-backend-cli delete -t retentionservice
        This will delete all the retentionservice items

    Delete all the services of an host from the backend:
        alignak-backend-cli delete -t service test_host/*
        This will delete all the services of the host named test_host

Hints and tips:
    You can operate on any backend endpoint: user, host, service, graphite, ... see␣
↪the
    Alignak backend documentation (http://alignak-backend.readthedocs.io/) to get a␣
↪full
    list of the available endpoints and their data fields.

    For a service specify the name as 'host_name/service_name' to get a service for a
    specific host, else the script will return the first serice with the required name

    By default, the script embeds in the provided result all the possible embeddable␣
↪data.
    As such, when you get a service, you will also get its host, check period, ...
    Unfortunately, the same embedding can not be used when adding or updating an item␣
↪:(

    Use the -m (--model) option to get the templates lists for the host, service or␣
↪user
    when you get a list. If not used, the list do not include the templates

    Use the -e (--embedded) option to get the linked objects embedded in the output.␣
↪For
    an host, as an example, the result will include the linked check period, contacts,
    check command,... If not used, the result will only include the linked objects␣
↪identifier.

    To get the list of all the services of an host, you can get the service list with
    a wildcard in the host name. For all the services of the host named 'passive-01',
    use 'passive-01/*' as in 'alignak-backend-cli get -l -t service passive-01/*'
```

(continues on next page)

(continued from previous page)

```
    To get all the information for an host, including the services, you can use
    a wildcard in the host name. For all the information of the host named 'passive-01
↪',
    use 'passive-01/*' as in 'alignak-backend-cli get -t host passive-01/*'. Using␣
↪the -e
    option will include all the related objects of the host and its services in the
    dump file.

    If somehow you need to update an item and post all the data when updating, use the
    `-i` option. This will use the data read from the backend and update this data␣
↪with
    the one provided in the data file specified in the `-d` option.

    Use the -v option to have more information
```

Note: this is not automatically updated from the source code. To get the most recent version, run *alignak-backend-cli -h*!

2.1.3 Some examples

The project repository folder *alignak_backend_client/examples* contains many example files built to be used or thanks to the *alignak-backend-cli* tool.

The following chapters give some command line examples. Do not hesitate to run *alignak-backend-cli -h* for the online help ;)

To have more information about the script execution, use the *-v / –verbose* parameter to activate the verbose mode. To test a command before executing operations in the backend, you can use the dry-run mode (*-c / –check*).

Note: the provided examples may not be up-to-date because of some recent Alignak backend data model modifications.

Add elements

To add an element (eg. an host named *host_name*) to the backend, run this command:

```
alignak-backend-cli -t host add host_name
```

This command will try to add an *host* element named *host_name* into the backend.

Note that most of the elements managed by the Alignak backend can be added without specifying any parameters. If some mandatory parameters are required, error message will be raised by the script and the missing parameters will be explained in the messages.

To add an element (eg. an host named *host_name*) to the backend, with some parameters, run this command:

```
alignak-backend-cli -t host -d data-file.json add host_name
```

This command will try to add an *host* element named *host_name* into the backend and it will use the parameters contained in the *data-file.json* file. As an example, the file *example_host_data.json* for common configuration parameters or *example_host_livestate.json* for an host live state update.

Note that most of the elements managed by the Alignak backend can be added without specifying any parameters. If some mandatory parameters are required, an error message will be raised by the script and the missing parameters will be explained in the messages.

To add a list of elements of a certain type (eg. a list of hosts) into the backend, run this command:

```
alignak-backend-cli -t host -d data-file.json add
```

Note that the command is the same as for adding an host but the host name is not present!

This command will try to add the *host* elements defined in the *data-file.json* file. This file must contain an array of *host*, each one having a defined *name* property. As an example, the file *example_host_data.json* for common configuration parameters or *example_host_livestate.json* for an host live state update.

Update elements

To update the elements in the Alignak backend you will use almost the same commands as for adding elements, except that you will use the *update* action word instead of the *add* action word. The provided json data file contains the definition of the properties to update for the targeted element. Using *-i / –include-read-data* argument will also update the existing properties that got read when the element was searched before being updated. In some rare situation, it may be necessary to use this argument...

This command will update an host named *host_name*:

```
alignak-backend-cli -t host -d data-file.json update host_name
```

Get elements

To get the list of all the elements of a certain type (eg. the list of all hosts) from the backend, run this command:

```
alignak-backend-cli -t host list
```

This will store the result in a file named *alignak-object-list-hosts.json* in the current directory. Adding the *-e* parameter to the command line will also get the linked elements for each element of the list (eg. the commands, timeperiods, ... linked to each host).

To get a specific element (eg. an host named *host_name*) from the backend, run this command:

```
alignak-backend-cli -t host get host_name
```

This will store the result in a file named *alignak-object-dump-host-host_name.json* in the current directory. As for the list, the *-e* parameter will embed the linked elements.

To get the list of all the services of an host from the backend, run this command:

```
alignak-backend-cli -t service list host_name/*
```

This will store the list of all the host *host_name* in a file.

To get an host and the list of all its services, run this command:

```
alignak-backend-cli -t host get host_name/*
```

This will store the list of all the host *host_name* in a file.

Delete elements

To delete the list of all the elements of a certain type (eg. the list of all hosts) from the backend, run this command:

```
alignak-backend-cli -t host delete
```

This will delete all the hosts defined in the backend!

To delete all the services related to an host (eg. an host named *host_name*) from the backend, run this command:

```
alignak-backend-cli -t service delete host_name/*
```

This will delete all the services linked to the host named *host_name* from the backend.

To delete a specific element (eg. an host named *host_name*) from the backend, run this command:

```
alignak-backend-cli -t host delete host_name
```

This will delete the host named *host_name* from the backend.

Beware that deleting some elements may create corrupted data in your Alignak backend! Deleting an host without having previously deleted its services will create orphan services that try to be linked to a non-existing host... take care of the deletion order: delete services of an host before the host!

2.1.4 An idea / some tests for the Alignak checks packs

The *alignak-backend-cli* tool allows to mainpulate Alignak backend data. This is the main driver that made an idea raise: why not updating the Alignak backend to install some checks packs?

With the *alignak-backend-cli*, it is easy to add new hosts/ services templates, new commands, new groups, ...

This is the procedure with the examples provided in the project repository folder *alignak_backend_client/examples*. Starting with an empty backend

```
# Add some commands and templates
alignak-backend-cli -v -t command -d examples/checks-pack-commands.json add
alignak-backend-cli -v -t user -d examples/checks-pack-users-templates.json add
alignak-backend-cli -v -t host -d examples/checks-pack-hosts-templates.json add
alignak-backend-cli -v -t service -d examples/checks-pack-services-templates.json add

# Use the Alignak Webui to check the backend content in a friendly manner;)
# Else:
alignak-backend-cli -v -t command list
alignak-backend-cli -v -t user list
alignak-backend-cli -v -t host -m list
alignak-backend-cli -v -t service -m list
# Use the '-m' argument for host and service to get the models (templates)
# Check the content of the alignak-object-list-*.json files in the current directory
# As examples, the corresponding files are present in the examples directory,␣
↪prefixed with checks-pack...

# Add an host from a template
alignak-backend-cli -v -t host -T windows-passive-host add host_test

# Add an host from a template and some more parameters
alignak-backend-cli -v -t host -d examples/example_host_data.json add host_test_2

# Get the hosts / services list
alignak-backend-cli -v -t host list
alignak-backend-cli -v -t service list
# As examples, the result files are present in the examples directory:
# - checks-pack-alignak-object-list-hosts.json
# - checks-pack-alignak-object-list-services.json
```

To restart from an empty backend:

```
$ mongo
$ use alignak-backend
$ db.dropDatabase()
$ Ctrl+C
$ alignak-backend-uwsgi
```

2.2 Python client documentation

Source code documentation:

2.2.1 Alignak REST backend client library

This module is a Python library used for connecting to an Alignak backend.

The *Backend* class implements the necessary methods to establish a connection and interact with the backend REST API.

Backend interaction will necessarily start with a *login* and end with a *logout*. In between, using the *get*, *post*, *patch* and *delete* functions will allow to manipulate the backend elements.

The Alignak backend data model is documented here.

2.2.2 Backend class

class alignak_backend_client.client.**Backend**(*endpoint*, *processes=1*)

Bases: object

Backend client class to communicate with an Alignak backend

Provide the backend endpoint URL to initialize the client (eg. http://127.0.0.1:5000)

static decode(*response*)

Decodes and returns the response as JSON (dict) or raise BackendException :param response: requests.response object :return: dict

delete(*endpoint*, *headers*)

Method to delete an item or all items

headers['If-Match'] must contain the _etag identifier of the element to delete

Parameters

- **endpoint** (*str*) – endpoint (API URL)
- **headers** (*dict*) – headers (example: Content-Type)

Returns response (deletion information)

Return type 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_all(*endpoint*, *params=None*)

Get all items in the specified endpoint of alignak backend

If an error occurs, a BackendException is raised.

If the max_results parameter is not specified in parameters, it is set to BACKEND_PAGINATION_LIMIT (backend maximum value) to limit requests number.

This method builds a response that always contains: _items and _status:

```
{
    u'_items': [
        ...
    ],
    u'_status': u'OK'
}
```

Parameters

- **endpoint** (*str*) – endpoint (API URL) relative from root endpoint
- **params** (*dict*) – list of parameters for the backend API

Returns dict of properties

Return type dict

get_domains()

Connect to alignak backend and retrieve all available child endpoints of root

If connection is successful, returns a list of all the resources available in the backend: Each resource is identified with its title and provides its endpoint relative to backend root endpoint.:

```
[
    {u'href': u'loghost', u'title': u'loghost'},
    {u'href': u'escalation', u'title': u'escalation'},
    ...
]
```

If an error occurs a BackendException is raised.

If an exception occurs, it is raised to caller.

Returns list of available resources

Return type list

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*, *generate='enabled'*, *proxies=None*)
Log into the backend and get the token

generate parameter may have following values: - enabled: require current token (default) - force: force new token generation - disabled

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
- **proxies** (*dict*) – dict of proxy (http and / or https)

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*, *headers=None*, *inception=False*)
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

put(*endpoint*, *data*, *headers=None*, *inception=False*)
Method to replace an item

The headers must include an If-Match containing the object _etag. headers = {'If-Match': contact_etag}

The data dictionary contain all fields.

If the puting 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 a new get request on the endpoint to refresh the _etag and then a new put 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 put response from the backend

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

2.2.3 BackendException class

exception alignak_backend_client.client.**BackendException**(*code*, *message*, *response=None*)

Bases: exceptions.Exception

Specific backend exception class. This specific exception is raised by the module when an error is encountered.

It provides an error code, an error message and the backend response.

Defined error codes:

- 1000: first stage error, exception raising between the client and the backend when connecting
- <1000: second stage error. Connection between client and backend is ok,

but the backend returns errors on requests

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

Index