fasjson-client

Release 1.0.8

Fedora Infrastructure

Feb 21, 2024

USAGE GUIDE

A python client library for the FASJSON API

This client uses the bravado library to build dynamic api methods based on open-api specs (version 2.0).

CHAPTER

ONE

USAGE

Instantiate the client with the FASJSON URL you want to use:

```
>>> from fasjson_client import Client
>>> c = Client('http://fasjson.example.com')
>>> c.whoami().result
{'dn': 'uid=admin,cn=users,cn=accounts,dc=example,dc=test', 'username': 'admin', 'service
↪': None, 'uri': 'http://fasjson.example.test/fasjson/v1/users/admin/'}
```

CHAPTER

TWO

AUTHENTICATION

Authentication is done with Kerberos. If you want to explicitely specify a principal to authenticate as, use the `principal` constructor argument:

```
c = Client('http://fasjson.example.com', principal='admin@EXAMPLE.TEST')
```

2.1 Configuring an application for Kerberos authentication

Users authenticate via `kinit`, applications authenticate via keytabs. It is highly recommended to use gssproxy in order to keep your keytabs secure.

- First, install gssproxy with `dnf install gssproxy`
- Create the service that you want to authenticate as in IPA: `ipa service-add SERVICE/host-fqdn` (for example `ipa service-add HTTP/server.example.com`)
- Get the keytab for that service and store it in gssproxy's directory: `ipa-getkeytab -p SERVICE/host-fqdn -k /var/lib/gssproxy/service.keytab` (for example `ipa-getkeytab -p HTTP/server.example.com -k /var/lib/gssproxy/httpd.keytab`)
- Add a configuration file for your service in gssproxy's configuration directory:

```
# /etc/gssproxy/50-servicename.conf

[service/servicename]
  mechs = krb5
  cred_store = keytab:/var/lib/gssproxy/service.keytab
  cred_store = client_keytab:/var/lib/gssproxy/service.keytab
  allow_constrained_delegation = true
  allow_client_ccache_sync = true
  cred_usage = both
  euid = user_the_service_runs_as
```

For example:

```
# /etc/gssproxy/80-httpd.conf

[service/httpd]
  mechs = krb5
  cred_store = keytab:/var/lib/gssproxy/httpd.keytab
  cred_store = client_keytab:/var/lib/gssproxy/httpd.keytab
  allow_constrained_delegation = true
```

(continues on next page)

(continued from previous page)

```
allow_client_ccache_sync = true
cred_usage = both
euid = apache
```

- Restart gssproxy with `systemctl restart gssproxy`
- Configure the service to run with the `GSS_USE_PROXY` environment variable set. Services started by systemd can be configured with a service configuration file, for example with the httpd service:

```
# /etc/systemd/system/httpd.service.d/gssproxy.conf
# /usr/lib/systemd/system/httpd.service.d/gssproxy.conf

[Service]
Environment=KRB5CCNAME=/tmp/krb5cc-httpd
Environment=GSS_USE_PROXY=yes
```

Your service should now be able to authenticate with Kerberos

CHAPTER

THREE

PAGINATION

Some operations can be paginated:

```
>>> from fasjson_client import Client
>>> c = Client('http://fasjson.example.com')
>>> response = c.list_users(page_size=2)
>>> response.result
[{'username': 'user1', [...]}, {'username': 'user2', [...]}]
```

The pagination data is available in the `page` property:

```
>>> response.page
{'total_results': 52, 'page_size': 2, 'page_number': 1, 'total_pages': 26}
```

Next and previous pages are available with the `next_page()` and `prev_page()` methods, which return the same class of objects:

```
>>> response.next_page().result
[{'username': 'user3', [...]}, {'username': 'user4', [...]}]
```

CHAPTER

FOUR

SELECTING ATTRIBUTES

You can select which attributes you want to get from the server using the `X-Fields` header. The header is given as a list of attribute names:

```
>>> from fasjson_client import Client
>>> c = Client('http://fasjson.example.com')
>>> response = c.list_users(
...     page_size=1,
...     _request_options={
...         "headers": {"X-Fields": ["username", "emails"]}
...     }
... )
>>> response.result
[{'username': 'user1', 'emails': ['user1@example.com']}]
```

CHAPTER

FIVE

COMMAND LINE

This package also provides a command-line client to do some operations. Install the dependencies with `poetry install -E cli` and run `fasjson-client --help` to see which operations are available.

CHAPTER

SIX

MIGRATING FROM PYTHON-FEDORA

Fasjson-client provides functionality for the most important endpoints previously exposed in python-fedora. Below is a list of common python-fedora endpoints and their fasjson-client alternatives.

6.1 Creating a client

The instantiation of a client can be done similarly to python-fedora, except there is no need to provide a username and password with which to authenticate. Instead, this authentication is performed by your service via Kerberos.

For more information, please see *Usage*. The content below assumes you have setup your client as detailed in *Usage*.

Pagination is supported in some of the `fasjson_client` API calls listed below, for more information about how to use it see *Pagination*.

The following sections are divided by the objects requested, and each corresponding python-fedora method is listed as a heading - with the appropriate fasjson-client endpoint then explained below.

6.2 Groups

6.2.1 group_by_id

You must now use the `groupname` instead of `id`.

```
>>> client.get_group(groupname="testGroup").result
{'groupname': 'testGroup', 'uri': 'http://fasjson.example.test/fasjson/v1/groups/
↪testGroup/'}
```

6.2.2 group_by_name

```
>>> client.get_group(groupname="testGroup").result
{'groupname': 'testGroup', 'uri': 'http://fasjson.example.test/fasjson/v1/groups/
↪testGroup/'}
```

6.2.3 group_members

```
>>> client.list_group_members(groupname="testGroup", page_size=5).result
[{'username': 'user1', [...]}, {'username': 'user2', [...]}]
```

6.3 People

6.3.1 person_by_id

You must now use the person's `username` instead of `id`

```
>>> client.get_user(username="test").result
{'username': 'test', 'surname': 'user', 'givenname': 'test', 'emails': ['test@example.
↪test'], 'ircnicks': ['test', 'test_1'], 'locale': 'en-US', 'timezone': None, 'gpgkeyids
↪': None, 'certificates': None, 'creation': None, 'locked': False, 'uri': 'http://
↪fasjson.example.test/fasjson/v1/users/test/'}
```

6.3.2 person_by_username

```
>>> client.get_user(username="test").result
{'username': 'test', 'surname': 'user', 'givenname': 'test', 'emails': ['test@example.
↪test'], 'ircnicks': ['test', 'test_1'], 'locale': 'en-US', 'timezone': None, 'gpgkeyids
↪': None, 'certificates': None, 'creation': None, 'locked': False, 'uri': 'http://
↪fasjson.example.test/fasjson/v1/users/test/'}
```

6.3.3 user_data

```
>>> client.list_users(page_size=50).result
[{'username': 'user1', [...]}, {'username': 'user2', [...]}]
```

6.3.4 people_by_groupname

```
>>> client.list_group_members(groupname="testGroup", page_size=5).result
[{'username': 'user1', [...]}, {'username': 'user2', [...]}]
```

6.4 Getting all items at once

The `list_all_entities` method is an iterator over all records of an entity in FASJSON, for example `users` or `groups`. They will be retrieved in multiple server calls (using pagination). You can specify the number of users that should be returned in each server call in the `page_size` argument if you have performance issues, but the default should be fine. An example with users:

```
>>> for user in client.list_all_entities("users", page_size=1000):
...     print(user)
{'username': 'user1', [...]}
{'username': 'user2', [...]}
{'username': 'user3', [...]}
[...]
```

CHAPTER

SEVEN

CONTRIBUTING

7.1 Development

Install dependencies:

```
poetry install
```

Run the tests:

```
tox
```

7.2 License

Licensed under lgpl-3.0