stashy Documentation

Release 0.1

Cosmin Stejerean

Nov 17, 2017

Contents

Python API client for the Atlassian Stash REST API

CHAPTER 1

Installation

```
pip install stashy
```

CHAPTER 2

Usage

```
import stashy
stash = stashy.connect("http://localhost:7990/stash", "admin", "admin")
```

CHAPTER 3

Examples

Retrieve all groups:

```
stash.admin.groups.list()
```

Retrieve all users that match a given filter:

```
stash.admin.users.list(filter="admin")
```

Add a user to a group:

```
stash.admin.groups.add_user('stash-users', 'admin')
```

Iterate over all projects (that you have access to):

```
stash.projects.list()
```

List all the repositories in a given project:

```
stash.projects[PROJECT].repos.list()
```

List all the commits in a pull request:

```
list(stash.projects[PROJECT].repos[REPO].pull_requests.commits())
```

CHAPTER 4

Contents:

4.1 API

stashy.**connect**(*url*, *username*, *password*, *verify=True*)
Connect to a Stash instance given a username and password.

This is only recommended via SSL. If you need are using self-signed certificates, you can use verify=False to ignore SSL verifcation.

class stashy.admin.users.**Users**(*url*, *client*, *parent*, *api_path=None*, *branches_api_path=None*)

add(*name*, *password*, *displayName*, *emailAddress*, *addToDefaultGroup=True*)
Add a user, returns a dictionary containing information about the newly created user

add_group(*user*, *group*)
Add the given user to the given user.

credentials(*name*, *new_password*)
Update a user's password.

delete(*user*)
Delete a user.

more_members(*user*, *filter=None*)
Retrieves a list of groups the specified user is a member of.

filter: if specified only groups with names containing the supplied string will be returned

more_non_members(*user*, *filter=None*)
Retrieves a list of groups that the specified user is not a member of

filter: if specified only groups with names containing the supplied string will be returned

remove_group(*user*, *group*)
Remove the given user from the given group.

update(*name*, *displayName=None*, *emailAddress=None*)
Update the user information, and return the updated user info.

None is used as a sentinel value, use empty string if you mean to clear.

class stashy.admin.groups.**Groups**(*url*, *client*, *parent*, *api_path=None*, *branches_api_path=None*)

add(*group*)
Add a group, returns a dictionary containing the group name

add_user(*group*, *user*)
Add a user to a group.

delete(*group*)
Delete a group.

more_members(*group*, *filter=None*)
Retrieves a list of users that are members of a specified group.

filter: return only users with usernames, display names or email addresses containing this string

more_non_members(*group*, *filter=None*)
Retrieves a list of users that are not members of a specified group.

filter: return only users with usernames, display names or email addresses containing this string

remove_user(*group*, *user*)
Remove a user to a group.

class stashy.permissions.**Users**(*url*, *client*, *parent*, *api_path=None*, *branches_api_path=None*)

grant(*user*, *permission*)
Promote or demote the permission level of a user.

Depending on context, you may use one of the following set of permissions:

global permissions:

- LICENSED_USER
- PROJECT_CREATE
- ADMIN
- SYS_ADMIN

project permissions:

- PROJECT_READ
- PROJECT_WRITE
- PROJECT_ADMIN

repository permissions:

- REPO_READ
- REPO_WRITE
- REPO_ADMIN

none(*filter=None*)
Retrieve users that have no granted permissions.

filter: if specified only user names containing the supplied string will be returned

revoke(*user*)
Revoke all permissions for a user.

class stashy.permissions.**Groups**(*url*, *client*, *parent*, *api_path=None*, *branches_api_path=None*)

grant(*group*, *permission*)
Promote or demote a user's permission level.

Depending on context, you may use one of the following set of permissions:

global permissions:

- LICENSED_USER
- PROJECT_CREATE
- ADMIN
- SYS_ADMIN

project permissions:

- PROJECT_READ
- PROJECT_WRITE
- PROJECT_ADMIN

none(*filter=None*)
Retrieve groups that have no granted permissions.

filter: return only group names containing the supplied string will be returned

revoke(*group*)
Revoke all permissions for a group.

class stashy.projects.**Projects**(*url*, *client*, *parent*, *api_path=None*, *branches_api_path=None*)

create(*key*, *name*, *description=''*, *avatar=None*)
Create a project. If supplied, avatar should be a base64 encoded image.

get(*project*)
Retrieve the project matching the supplied key.

class stashy.projects.**Project**(*key*, *url*, *client*, *parent*)

delete()
Delete the project

update(*new_key=None*, *name=None*, *description=None*, *avatar=None*, *public=None*)
Update project information. If supplied, avatar should be a base64 encoded image.

None is used as a sentinel so use '' to clear a value.

class stashy.repos.**Repos**(*url*, *client*, *parent*, *api_path=None*, *branches_api_path=None*)

create(*name*, *scmId='git'*, *forkable=True*)
Create a repository with the given name

class stashy.repos.**Repository**(*slug*, *url*, *client*, *parent*)

branches(*filterText=None*, *orderBy=None*, *details=None*)
Retrieve the branches matching the supplied filterText param.

browse(*path=''*, *at=None*, *type=False*, *blame=''*, *noContent=''*)
Retrieve a page of content for a file path at a specified revision.

changes(*until*, *since=None*)
Retrieve a page of changes made in a specified commit.

since: the changeset to which until should be compared to produce a page of changes. If not specified the parent of the until changeset is used.

until: the changeset to retrieve file changes for.

commits(*until*, *since=None*, *path=None*)
Retrieve a page of changesets from a given starting commit or between two commits. The commits may be identified by hash, branch or tag name.

since: the changeset id or ref (exclusively) to restrieve changesets after until: the changeset id or ref (inclusively) to retrieve changesets before. path: an optional path to filter changesets by.

Support for withCounts is not implement.

default_branch
Get or set the default branch

delete()
Schedule the repository to be deleted

files(*path=''*, *at=None*)
Retrieve a page of files from particular directory of a repository. The search is done recursively, so all files from any sub-directory of the specified directory will be returned.

fork(*name=None*, *project=None*)
Fork the repository.

name - Specifies the forked repository's name Defaults to the name of the origin repository if not specified

project - Specifies the forked repository's target project by key Defaults to the current user's personal project if not specified

forks()
Retrieve repositories which have been forked from this one.

get()
Retrieve the repository

get_all_branches(*items*)
Return list of all branches in this project and the repository :param items: limit parameter (max items in result) :return:

get_commit(*commit*)
Returns detailed information about a given commit :param commit: like "1c972ea39318a4b3ce99bc51ab03277138c586ea" :return:

tags(*filterText=None*, *orderBy=None*)
Retrieve the tags matching the supplied filterText param.

update(*name*)
Update the name of a repository.

The repository's slug is derived from its name. If the name changes the slug may also change.

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Python Module Index

S

Index