Gerrit Client with Python

Release 1.0.1

Jialiang Shi

Aug 01, 2023

CONTENTS

python-gerrit-api provides a simple interface for clients to interact with Gerrit via the REST API.

CHAPTER

ONE

COMPATIBILITY

- This package is compatible Python versions 2.7, 3.5+.

CHAPTER

TWO

INSTALLATION

Use **pip** to install the latest stable version of `python-gerrit-api`:

```
$ sudo pip install python-gerrit-api
```

The current development version is available on github. Use **git** and **python setup.py** to install it:

```
$ git clone https://github.com/shijl0925/python-gerrit-api.git
$ cd python-gerrit-api
$ sudo python setup.py install
```

CHAPTER

THREE

CHANGE LOG

See the CHANGELOG.md file.

CHAPTER

FOUR

DOCUMENTATION

This part of the documentation will show you how to get started in using python-gerrit-api.

The Client is easy to use, you just need to initialize it with the connection parameters.

4.1 Setup a Gerrit Client

```
from gerrit import GerritClient

client = GerritClient(base_url="https://yourgerrit", username='******', password='xxxxx')
```

4.2 Example

Refer to the example script for a full working example.

4.2.1 API examples

api/projects

Examples

setup gerrit client:

```
from gerrit import GerritClient
client = GerritClient(base_url="https://yourgerrit", username='******', password='xxxxx')
```

Lists the projects:

```
projects = client.projects.list()
```

Retrieves a project.:

```
project = client.projects.get(project_name="MyProject")
```

Creates a new project.:

```
input_ = {
    "description": "This is a demo project.",
    "submit_type": "INHERIT",
    "owners": [
      "MyProject-Owners"
    ]
}
project = client.projects.create('MyProject', input_)
```

Delete the project, requires delete-project plugin:

```
client.projects.delete("MyProject")
# or
project = client.projects.get(project_name="MyProject")
project.delete()
```

api/project

Examples

setup gerrit client and retrieve one project instance:

```
from gerrit import GerritClient
client = GerritClient(base_url="https://yourgerrit", username='*******', password='xxxxx')

project = client.projects.get('python-sonarqube-api')
```

Retrieves the description of a project.:

```
description = project.get_description()
```

Sets the description of a project.:

```
input_ = {
    "description": "Plugin for Gerrit that handles the replication.",
    "commit_message": "Update the project description"
}
result = project.set_description(input_)
```

Deletes the description of a project.:

```
project.delete_description()
```

Delete the project, requires delete-project plugin:

```
project.delete()
```

Sets the configuration of a project.:

```
input_ = {
    "description": "demo project",
    "use_contributor_agreements": "FALSE",
```

(continues on next page)

(continued from previous page)

```
    "use_content_merge": "INHERIT",
    "use_signed_off_by": "INHERIT",
    "create_new_change_for_all_not_in_target": "INHERIT",
    "enable_signed_push": "INHERIT",
    "require_signed_push": "INHERIT",
    "reject_implicit_merges": "INHERIT",
    "require_change_id": "TRUE",
    "max_object_size_limit": "10m",
    "submit_type": "REBASE_IF_NECESSARY",
    "state": "ACTIVE"
}
result = project.set_config(input_)
```

Lists the access rights for a single project.:

```
access_rights = project.get_access_rights()
```

Create Change for review. support this method since v3.3.0:

```
input_ = {
    "subject": "Let's support 100% Gerrit workflow direct in browser",
    "branch": "stable",
    "topic": "create-change-in-browser",
    "status": "NEW"
}

result = project.create_change(input_)
```

List the branches of a project.:

```
branches = project.branches.list()
```

get a branch by ref:

```
branch = project.branches.get('refs/heads/stable')
```

Creates a new branch.:

```
input_ = {
    'revision': '76016386a0d8ecc7b6be212424978bb45959d668'
}
new_branch = project.branches.create('stable', input_)
```

Delete a branch.:

```
branch.delete()
```

Retrieves a commit of a project.:

```
commit = project.get_commit('c641ab4dd180b4184f2663bd28277aa796b36417')
```

Retrieves the branches and tags in which a change is included.:

```
result = commit.get_include_in()
```

Gets the content of a file from a certain commit.:

```
content = commit.get_file_content('sonarqube/community/components.py')
```

Cherry-picks a commit of a project to a destination branch.:

```
input_ = {
    "message": "Test Cherry Pick",
    "destination": "stable"
}

commit.cherry_pick(input_)
```

Lists the files that were modified, added or deleted in a commit.:

```
result = commit.list_change_files()
```

api/changes

Examples

setup gerrit client:

```
from gerrit import GerritClient
client = GerritClient(base_url="https://yourgerrit", username='*******', password='xxxxx')
```

Queries changes.:

```
result = client.changes.search(query=['status:open'])
# or
query = ["is:open+owner:self", "is:open+reviewer:self+-owner:self",
↪"is:closed+owner:self+limit:5"]
result = client.changes.search(query=query, options=["LABELS"])
```

Retrieves a change.:

```
change = client.changes.get("MyProject~master~I39b027b763fb0b0dc7ed6c9e6bb5128d882dbe7c")
```

create a change.:

```
input_ = {
    "project": "myProject",
    "subject": "Let's support 100% Gerrit workflow direct in browser",
    "branch": "stable",
    "topic": "create-change-in-browser",
    "status": "NEW"
}
result = client.changes.create(input_)
```

Deletes a change.:

```
client.changes.delete("MyProject~master~I39b027b763fb0b0dc7ed6c9e6bb5128d882dbe7c")
# or
change = client.changes.get("MyProject~master~I39b027b763fb0b0dc7ed6c9e6bb5128d882dbe7c")
change.delete()
```

api/change

Examples

setup gerrit client and retrieve one change instance:

```
from gerrit import GerritClient
client = GerritClient(base_url="https://yourgerrit", username='******', password='xxxxx')

change = client.changes.get(
    "MyProject~master~I39b027b763fb0b0dc7ed6c9e6bb5128d882dbe7c"
)
```

Update an existing change by using a MergePatchSetInput entity.:

```
input_ = {
    "subject": "Merge master into stable",
    "merge": {
      "source": "refs/heads/master"
    }
}

result = change.update(input_)
```

Creates a new patch set with a new commit message.:

```
input_ = {
    "message": "New Commit message \\n\\nChange-Id:␣
↪I10394472cbd17dd12454f229e4f6de00b143a444\\n"
}

result = change.set_commit_message(input_)
```

Retrieves the topic of a change.:

```
topic = change.get_topic()
```

Sets the topic of a change.:

```
topic = change.set_topic("test topic")
```

Deletes the topic of a change.:

```
change.delete_topic()
```

Retrieves the account of the user assigned to a change.:

```
assignee = change.get_assignee()
```

Sets the assignee of a change.:

```
input_ = {
    "assignee": "jhon.doe"
}
result = change.set_assignee(input_)
```

Returns a list of every user ever assigned to a change, in the order in which they were first assigned.:

result = change.get_past_assignees()

Deletes the assignee of a change.:

```
result = change.delete_assignee()
```

Abandons a change.:

```
result = change.abandon()
```

Restores a change.:

```
result = change.restore()
```

Deletes a change.:

```
change.delete()
```

Marks the change to be private. Only open changes can be marked private.:

```
input_ = {
    "message": "After this security fix has been released we can make it public now."
}
change.mark_private(input_)
```

Marks the change to be non-private. Note users can only unmark own private changes.:

```
input_ = {
    "message": "This is a security fix that must not be public."
}
change.unmark_private(input_)
# or
change.unmark_private()
```

get one revision by revision id.:

```
revision = change.get_revision("534b3ce21655a092eccf72680f2ad16b8fecf119")
```

4.3 API Reference

If you are looking for information on a specific function, class or method, this part of the documentation is for you.

4.3.1 gerrit package

Subpackages

gerrit.accounts package

Submodules

gerrit.accounts.account module

class gerrit.accounts.account.**GerritAccount**(*account*, *gerrit*)

Bases: *GerritBase*

check_capability(*capability*)

Checks if a user has a certain global capability.

Parameters
capability –

Returns

delete_active()

Sets the account state to inactive. If the account was already inactive the response is '409 Conflict'.

Returns

delete_draft_comments(*input_*)

Deletes some or all of a user's draft comments.

```
input_ = {
    "query": "is:abandoned"
}
account = client.accounts.get('kevin.shi')
result = account.delete_draft_comments(input_)
```

Parameters
input – the DeleteDraftCommentsInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-accounts.html#delete-draft-comments-input

Returns

delete_external_ids(*input_*)

Delete a list of external ids for a user account. Only external ids belonging to the caller may be deleted. Users that have Modify Account can delete external ids that belong to other accounts.

```
input_ = [
    "mailto:john.doe@example.com"
]
```

(continues on next page)

(continued from previous page)

```
account = client.accounts.get('kevin.shi')
result = account.delete_external_ids(input_)
```

Parameters
input – the external ids as list

Returns

delete_http_password()

Deletes the HTTP password of an account.

Returns

delete_name()

Deletes the name of an account. Some realms may not allow to delete the account name. In this case the request is rejected with '405 Method Not Allowed'.

Returns

delete_watched_projects(*input_*)

Projects posted to this endpoint will no longer be watched.

```
input_ = [
    {
        "project": "Test Project 1",
        "filter": "branch:master"
    }
]

account = client.accounts.get('kevin.shi')
result = account.delete_watched_projects(input_)
```

Parameters
input – the watched projects as list

Returns

property emails

get_active()

Checks if an account is active.

Returns

get_avatar()

Retrieves the avatar image of the user, requires avatars-gravatar plugin.

Returns

get_avatar_change_url()

Retrieves the avatar image of the user, requires avatars-gravatar plugin.

Returns

get_default_starred_changes()

Gets the changes that were starred with the default star by the identified user account.

Returns

get_detail()

fetch account info in more details, such as: registered_on

Returns

get_diff_preferences()

Retrieves the diff preferences of a user.

Returns

get_edit_preferences()

Retrieves the edit preferences of a user.

Returns

get_external_ids()

Retrieves the external ids of a user account. Only external ids belonging to the caller may be requested. Users that have Modify Account can request external ids that belong to other accounts.

Returns

get_name()

Retrieves the full name of an account.

Returns

get_oauth_token()

Returns a previously obtained OAuth access token. If there is no token available, or the token has already expired, '404 Not Found' is returned as response. Requests to obtain an access token of another user are rejected with '403 Forbidden'.

Returns

get_star_labels_from_change(*id_*)

Get star labels from a change.

Parameters
id – change id

Returns

get_starred_changes()

Gets the changes that were starred with any label by the identified user account.

Returns

get_status()

Retrieves the status of an account. If the account does not have a status an empty string is returned.

Getter
Retrieves the status of an account.

Setter
Sets the status of an account

Returns

get_user_preferences()

Retrieves the user's preferences.

Returns

get_watched_projects()

Retrieves all projects a user is watching.

Returns

property gpg_keys

property groups

Lists all groups that contain the specified user as a member.

Returns

index()

Adds or updates the account in the secondary index.

Returns

list_capabilities()

Returns the global capabilities that are enabled for the specified user.

Returns

list_contributor_agreements()

Gets a list of the user's signed contributor agreements.

Returns

modify_watched_projects(*input_*)

Add new projects to watch or update existing watched projects. Projects that are already watched by a user will be updated with the provided configuration.

```
input_ = [
    {
        "project": "Test Project 1",
        "notify_new_changes": true,
        "notify_new_patch_sets": true,
        "notify_all_comments": true,
    }
]

account = client.accounts.get('kevin.shi')
result = account.modify_watched_projects(input_)
```

Parameters

input – the ProjectWatchInfo entities as list

Returns

put_default_star_on_change(*id_*)

Star a change with the default label.

Parameters

id – change id

Returns

remove_default_star_from_change(*id_*)

Remove the default star label from a change. This stops notifications.

Parameters
id – change id

Returns

set_active()

Sets the account state to active.

Returns

set_diff_preferences(*input_*)

Sets the diff preferences of a user.

```
input_ = {
    "context": 10,
    "theme": "ECLIPSE",
    "ignore_whitespace": "IGNORE_ALL",
    "intraline_difference": true,
    "line_length": 100,
    "cursor_blink_rate": 500,
    "show_line_endings": true,
    "show_tabs": true,
    "show_whitespace_errors": true,
    "syntax_highlighting": true,
    "tab_size": 8,
    "font_size": 12
}

account = client.accounts.get('kevin.shi')
result = account.set_diff_preferences(input_)
```

Parameters
input – the DiffPreferencesInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-accounts.html#diff-preferences-input

Returns

set_displayname(*input_*)

Sets the display name of an account. support this method since v3.2.0

```
input_ = {
    "display_name": "Kevin"
}

account = client.accounts.get('kevin.shi')
result = account.set_displayname(input_)
```

Parameters
input – the DisplayNameInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-accounts.html#display-name-input

Returns

set_edit_preferences(*input_*)

Sets the edit preferences of a user.

```
input_ = {
    "theme": "ECLIPSE",
    "key_map_type": "VIM",
    "tab_size": 4,
    "line_length": 80,
    "indent_unit": 2,
    "cursor_blink_rate": 530,
    "hide_top_menu": true,
    "show_tabs": true,
    "show_whitespace_errors": true,
    "syntax_highlighting": true,
    "hide_line_numbers": true,
    "match_brackets": true,
    "line_wrapping": false,
    "auto_close_brackets": true
}

account = client.accounts.get('kevin.shi')
result = account.set_edit_preferences(input_)
```

Parameters

input – the EditPreferencesInfo entity, https://gerrit-review.googlesource.com/Documentation/rest-api-accounts.html#edit-preferences-info

Returns

set_http_password(*input_*)

Sets/Generates the HTTP password of an account.

```
input_ = {
    "generate": 'true',
    "http_password": "the_password"
}

account = client.accounts.get('kevin.shi')
result = account.set_http_password(input_)
```

Parameters

input – the HttpPasswordInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-accounts.html#http-password-input

Returns

set_name(*input_*)

Sets the full name of an account. Some realms may not allow to modify the account name. In this case the request is rejected with '405 Method Not Allowed'.

```
input_ = {
    "name": "Keven Shi"
}
```

(continues on next page)

(continued from previous page)

```
account = client.accounts.get('kevin.shi')
result = account.set_name(input_)
```

Parameters

input – the AccountNameInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-accounts.html#account-name-input

Returns

set_status(*status*)

Sets the status of an account.

Parameters

status – account status

Returns

set_user_preferences(*input_*)

Sets the user's preferences.

```
input_ = {
    "changes_per_page": 50,
    "show_site_header": true,
    "use_flash_clipboard": true,
    "expand_inline_diffs": true,
    "download_command": "CHECKOUT",
    "date_format": "STD",
    "time_format": "HHMM_12",
    "size_bar_in_change_table": true,
    "review_category_strategy": "NAME",
    "diff_view": "SIDE_BY_SIDE",
    "mute_common_path_prefixes": true,
}

account = client.accounts.get('kevin.shi')
result = account.set_user_preferences(input_)
```

Parameters

input – the PreferencesInput entity https://gerrit-review.googlesource.com/Documentation/rest-api-accounts.html#preferences-input

Returns

set_username(*input_*)

Sets the username of an account. Some realms may not allow to modify the account username. In this case the request is rejected with '405 Method Not Allowed'.

```
input_ = {
    "username": "shijl0925.shi"
}

account = client.accounts.get('kevin.shi')
result = account.set_username(input_)
```

Parameters
input – the UsernameInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-accounts.html#username-input

Returns

sign_contributor_agreement(*input_*)

Signs a contributor agreement.

```
input_ = {
    "name": "Individual"
}
account = client.accounts.get('kevin.shi')
result = account.sign_contributor_agreement(input_)
```

Parameters
input – the ContributorAgreementInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-accounts.html#contributor-agreement-input

Returns

property ssh_keys

update_star_labels_on_change(*id_*, *input_*)

Update star labels on a change.

```
input_ = {
    "add": ["blue", "red"],
    "remove": ["yellow"]
}

account = client.accounts.get('kevin.shi')
change_id = "myProject~master~I8473b95934b5732ac55d26311a706c9c2bde9940"
result = account.update_star_labels_on_change(change_id, input_)
```

Parameters

- **id** – change id
- **input** – the StarsInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-accounts.html#stars-input

Returns

gerrit.accounts.accounts module

class gerrit.accounts.accounts.**GerritAccounts**(*gerrit*)

Bases: object

create(*username*, *input_*)

Creates a new account.

```
input_ = {
    "name": "John Doe",
    "email": "john.doe@example.com",
    "ssh_key": "ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEA0T...YImydZAw==",
    "http_password": "19D9aIn7zePb",
    "groups": [
      "MyProject-Owners"
    ]
}
new_account = client.accounts.create('john.doe', input_)
```

Parameters

- **username** – account username
- **input** – the AccountInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-accounts.html#account-input

Returns

get(*account*)

Returns an account

Parameters
account – username or email or _account_id or 'self'

Returns

search(*query: str*, *limit: int = 25*, *skip: int = 0*, *detailed: bool = False*, *suggested: bool = False*, *all_emails: bool = False*)

Queries accounts visible to the caller.

Parameters

- **query** – Query string
- **limit** – Int value that allows to limit the number of accounts to be included in the output results
- **skip** – Int value that allows to skip the given number of accounts from the beginning of the list
- **detailed** – boolean value, if True then full name, preferred email, username and avatars for each account will be added to the output result
- **suggested** – boolean value, if True get account suggestions based on query string. If a result limit n is not specified, then the default 10 is used.
- **all_emails** – boolean value, if True then all registered emails for each account will be added to the output result

Returns

gerrit.accounts.emails module

class gerrit.accounts.emails.**GerritAccountEmail**(*email*, *account*, *gerrit*)

Bases: *GerritBase*

delete()

Deletes an email address of an account.

Returns

set_preferred()

Sets an email address as preferred email address for an account.

Returns

class gerrit.accounts.emails.**GerritAccountEmails**(*account*, *gerrit*)

Bases: object

create(*email*)

Registers a new email address for the user.

Returns

delete(*email*)

Deletes an email address of an account.

Parameters
email – account email

Returns

get(*email*)

Retrieves an email address of a user.

Returns

list()

Returns the email addresses that are configured for the specified user.

Returns

set_preferred(*email*)

Sets an email address as preferred email address for an account.

Parameters
email – account email

Returns

gerrit.accounts.gpg_keys module

class gerrit.accounts.gpg_keys.**GerritAccountGPGKey**(*id*, *account*, *gerrit*)

Bases: *GerritBase*

delete()

Deletes a GPG key of a user.

Returns

class gerrit.accounts.gpg_keys.**GerritAccountGPGKeys**(*account*, *gerrit*)

Bases: object

delete(*id_*)

Deletes a GPG key of a user.

Parameters
id – GPG key id

Returns

get(*id_*)

Retrieves a GPG key of a user.

Parameters
id – GPG key id

Returns

list()

Returns the GPG keys of an account.

Returns

modify(*input_*)

Add or delete one or more GPG keys for a user.

```
input_ = {
    "add": [
      "-----BEGIN PGP PUBLIC KEY BLOCK-----\n
      Version: GnuPG v1\n\n
      mQENBFXUpNcBCACv4paCiyKxZ0EcKy8VaWVNkJlNebRBiyw9WxU85wPOq5Gz/3GT\n
      RQwKqeY0SxVdQT8VNBw2sBe2m6eqcfZ2iKmesSlbXMe15DA7k8Bg4zEpQ0tXNG1L\n
      hceZDVQ1Xk06T2sgkunaiPsXi82nwN3UWYtDXxX4is5e6xBNL48Jgz4lbqo6+8D5\n
      vsVYiYMx4AwRkJyt/oA3IZAtSlY8Yd445nY14VPcnsGRwGWTLyZv9gxKHRUppVhQ\n
      E3o6ePXKEVgmONnQ4CjqmkGwWZvjMF2EPtAxvQLAuFa8Hqtkq5cgfgVkv/Vrcln4\n
      nQZVoMm3a3f5ODii2tQzNh6+7LL1bpqAmVEtABEBAAG0H0pvaG4gRG9lIDxqb2hu\n
      LmRvZUBleGFtcGxlLmNvbT6JATgEEwECACIFAlXUpNcCGwMGCwkIBwMCBhUIAgkK\n
      CwQWAgMBAh4BAheAAAoJEJNQnkuvyKSbfjoH/2OcSQOu1kJ20ndjhgY2yNChm7gd\n
      tU7TEBbB0TsLeazkrrLtKvrpW5+CRe07ZAG9HOtp3DikwAyrhSxhlYgVsQDhgB8q\n
      G0tYiZtQ88YyYrncCQ4hwknrcWXVW9bK3V4ZauxzPv3ADSloyR9tMURw5iHCIeL5\n
      fIw/pLvA3RjPMx4Sfow/bqRCUELua39prGw5Tv8a2ZRFbj2sgP5j8lUFegyJPQ4z\n
      tJhe6zZvKOzvIyxHO8llLmdrImsXRL9eqroWGs0VYqe6baQpY6xpSjbYK0J5HYcg\n
      TO+/u80JI+ROTMHE6unGp5Pgh/xIz6Wd34E0lWL1eOyNfGiPLyRWn1d0yZO5AQ0E\n
      VdSk1wEIALUycrH2HK9zQYdR/KJo1yJJuaextLWsYYn881yDQo/p06U5vXOZ28lG\n
      Aq/Xs96woVZPbgME6FyQzhf20Z2sbr+5bNo3OcEKaKX3Eo/sWwSJ7bXbGLDxMf4S\n
      etfY1WDC+4rTqE30JuC++nQviPRdCcZf0AEgM6TxVhYEMVYwV787YO1IH62EBICM\n
      SkIONOfnusNZ4Skgjq9OzakOOpROZ4tki5cH/5oSDgdcaGPy1CFDpL9fG6er2zzk\n
      sw3qCbraqZrrlgpinWcAduiao67U/dV18O6OjYzrt33fTKZ0+bXhk1h1gloC21MQ\n
      ya0CXlnfR/FOQhvuK0RlbR3cMfhZQscAEQEAAYkBHwQYAQIACQUCVdSk1wIbDAAK\n
      CRCTUJ5Lr8ikm8+QB/4uE+AlvFQFh9W8koPdfk7CJF7wdgZZ2NDtktvLL71WuMK8\n
      POmf9f5JtcLCX4iJxGzcWogAR5ed20NgUoHUg7jn9Xm3fvP+kiqL6WqPhjazd89h\n
      k06v9hPE65kp4wb0fQqDrtWfP1lFGuh77rQgISt3Y4QutDl49vXS183JAfGPxFxx\n
      8FgGcfNwL2LVObvqCA0WLqeIrQVbniBPFGocE3yA/0W9BB/xtolpKfgMMsqGRMeu\n
      9oIsNxB2oE61OsqjUtGsnKQi8k5CZbhJaql4S89vwS+efK0R+mo+0N55b0XxRlCS\n
      faURgAcjarQzJnG0hUps2GNO/+nM7UyyJAGfHlh5\n
      =EdXO\n
```

(continues on next page)

(continued from previous page)

```
      -----END PGP PUBLIC KEY BLOCK-----\n"
    ],
    "delete": [
      "DEADBEEF",
    ]
}
account = client.accounts.get('kevin.shi')
result = account.gpg_keys.modify(input_)
```

Parameters
input – the GpgKeysInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-accounts.html#gpg-keys-input

Returns

gerrit.accounts.ssh_keys module

class gerrit.accounts.ssh_keys.**GerritAccountSSHKey**(*seq*, *account*, *gerrit*)

Bases: *GerritBase*

delete()

Deletes an SSH key of a user.

Returns

class gerrit.accounts.ssh_keys.**GerritAccountSSHKeys**(*account*, *gerrit*)

Bases: object

add(*ssh_key*)

Adds an SSH key for a user. The SSH public key must be provided as raw content in the request body.

Parameters
ssh_key – SSH key raw content

Returns

delete(*seq*)

Deletes an SSH key of a user.

Parameters
seq – SSH key id

Returns

get(*seq*)

Retrieves an SSH key of a user.

Parameters
seq – SSH key id

Returns

list()

Returns the SSH keys of an account.

Returns

Module contents

gerrit.changes package

Subpackages

gerrit.changes.revision package

Submodules

gerrit.changes.revision.comments module

class gerrit.changes.revision.comments.**GerritChangeRevisionComment**(*id: str*, *change: str*, *revision: str*, *gerrit*)

Bases: *GerritBase*

delete(*input_=None*)

Deletes a published comment of a revision. Instead of deleting the whole comment, this endpoint just replaces the comment's message with a new message, which contains the name of the user who deletes the comment and the reason why it's deleted.

```
input_ = {
    "reason": "contains confidential information"
}

change = client.changes.get('Project~stable~
↪I10394472cbd17dd12454f229e4f6de00b143a444')
revision = change.get_revision('3848807f587dbd3a7e61723bbfbf1ad13ad5a00a')
comment = revision.comments.get("e167e775_e069567a")
result = comment.delete(input_)
# or
result = comment.delete()
```

Parameters

input – the DeleteCommentInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-changes.html#delete-comment-input

Returns

class gerrit.changes.revision.comments.**GerritChangeRevisionComments**(*change*, *revision*, *gerrit*)

Bases: `object`

get(*id_*)

Retrieves a published comment of a revision.

Parameters

id –

Returns

list()

Lists the published comments of a revision.

Returns

gerrit.changes.revision.drafts module

class gerrit.changes.revision.drafts.**GerritChangeRevisionDraft**(*id: str*, *change: str*, *revision: str*, *gerrit*)

Bases: *GerritBase*

delete()

Deletes a draft comment from a revision.

Returns

update(*input_*)

Updates a draft comment on a revision.

```
input_ = {
    "path": "sonarqube/cloud/duplications.py",
    "line": 25,
    "message": "[nit] trailing whitespace"
}
change = client.changes.get('Project~stable~
↪I10394472cbd17dd12454f229e4f6de00b143a444')
revision = change.get_revision('3848807f587dbd3a7e61723bbfbf1ad13ad5a00a')
draft = revision.drafts.get('89f04e8c_9b7fd51d')
result = draft.update(input_)
```

Parameters

input – the CommentInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-changes.html#comment-input

Returns

class gerrit.changes.revision.drafts.**GerritChangeRevisionDrafts**(*change*, *revision*, *gerrit*)

Bases: object

create(*input_*)

Creates a draft comment on a revision.

```
input_ = {
    "path": "sonarqube/cloud/duplications.py",
    "line": 15,
    "message": "[nit] trailing whitespace"
}
change = client.changes.get('Project~stable~
↪I10394472cbd17dd12454f229e4f6de00b143a444')
revision = change.get_revision('3848807f587dbd3a7e61723bbfbf1ad13ad5a00a')
new_draft = revision.drafts.create(input_)
```

Parameters

input – the CommentInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-changes.html#comment-input

Returns

delete(*id_*)

Deletes a draft comment from a revision.

Parameters
id – the draft comment id

Returns

get(*id_*)

Retrieves a draft comment of a revision that belongs to the calling user.

Parameters
id – the draft comment id

Returns

list()

Lists the draft comments of a revision that belong to the calling user.

Returns

gerrit.changes.revision.files module

class gerrit.changes.revision.files.**GerritChangeRevisionFile**(*path: str*, *json: dict*, *change: str*, *revision: str*, *gerrit*)

Bases: object

delete_reviewed()

Deletes the reviewed flag of the calling user from a file of a revision.

Returns

download_content()

Downloads the content of a file from a certain revision, in a safe format that poses no risk for inadvertent execution of untrusted code.

If the content type is defined as safe, the binary file content is returned verbatim. If the content type is not safe, the file is stored inside a ZIP file, containing a single entry with a random, unpredictable name having the same base and suffix as the true filename. The ZIP file is returned in verbatim binary form.

Returns

get_blame()

Gets the blame of a file from a certain revision.

Returns

get_content(*decode=False*)

Gets the content of a file from a certain revision. The content is returned as base64 encoded string.

Parameters
decode – Decode bas64 to plain text.

Returns

get_diff(*intraline=False*)

Gets the diff of a file from a certain revision.

Parameters
intraline – If the intraline parameter is specified, intraline differences are

included in the diff. :return:

set_reviewed()

Marks a file of a revision as reviewed by the calling user.

Returns

to_dict()

class gerrit.changes.revision.files.**GerritChangeRevisionFiles**(*change*, *revision*, *gerrit*)

Bases: object

get(*path*)

get a file by path

Parameters
path – file path

Returns

iterkeys()

Iterate over the paths of all files

Returns

keys()

Return a list of the file paths

Returns

poll()

Returns

search(*reviewed: Optional[bool] = None*, *base: Optional[int] = None*, *q: Optional[str] = None*, *parent: Optional[int] = None*)

Lists the files that were modified, added or deleted in a revision. The reviewed, base, q, and parent are mutually exclusive. That is, only one of them may be used at a time.

Parameters

- **reviewed** – return a list of the paths the caller has marked as reviewed
- **base** – return a map of the files which are different in this commit compared to the given revision. The revision must correspond to a patch set in the change.
- **q** – return a list of all files (modified or unmodified) that contain that substring in the path name.
- **parent** – For merge commits only, the integer-valued request parameter changes the response to return a map of the files which are different in this commit compared to the given parent commit.

Returns

Module contents

Submodules

gerrit.changes.change module

class gerrit.changes.change.**GerritChange**(*id: str*, *gerrit*)

Bases: *GerritBase*

abandon()

Abandons a change. Abandoning a change also removes all users from the attention set. If the change cannot be abandoned because the change state doesn't allow abandoning of the change, the response is "409 Conflict" and the error message is contained in the response body.

Returns

add_to_attention_set(*input_*)

Adds a single user to the attention set of a change. support this method since v3.3.0

A user can only be added if they are not in the attention set. If a user is added while already in the attention set, the request is silently ignored.

```
input_ = {
    "user": "John Doe",
    "reason": "reason"
}
change = client.changes.get('Project~stable~
↪I10394472cbd17dd12454f229e4f6de00b143a444')
result = change.add_to_attention_set(input_)
```

Parameters

input – the AttentionSetInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-changes.html#attention-set-input

Returns

check_submit_requirement(*input_*)

Tests a submit requirement.

Parameters

input – the SubmitRequirementInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-changes.html#submit-requirement-input

Returns

consistency_check()

Performs consistency checks on the change, and returns a ChangeInfo entity with the problems field set to a list of ProblemInfo entities.

Returns

create_empty_edit()

Creates empty change edit

Returns

create_merge_patch_set(*input_*)

Update an existing change by using a MergePatchSetInput entity. Gerrit will create a merge commit based on the information of MergePatchSetInput and add a new patch set to the change corresponding to the new merge commit.

```
input_ = {
    "subject": "Merge master into stable",
    "merge": {
      "source": "refs/heads/master"
    }
}

change = client.changes.get('Project~stable~
↪I10394472cbd17dd12454f229e4f6de00b143a444')
result = change.update(input_)
```

Parameters

input – the MergePatchSetInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-changes.html#merge-patch-set-input

Returns

delete()

Deletes a change.

Returns

delete_assignee()

Deletes the assignee of a change.

Returns

delete_topic()

Deletes the topic of a change.

Returns

delete_vote(*account*, *label*, *input_=None*)

Deletes a single vote from a change. Note, that even when the last vote of a reviewer is removed the reviewer itself is still listed on the change. If another user removed a user's vote, the user with the deleted vote will be added to the attention set.

```
input_ = {
    "notify": "NONE"
}

change = client.changes.get('Project~stable~
↪I10394472cbd17dd12454f229e4f6de00b143a444')
change.delete_vote('John', 'Code-Review', input_)
# or
change.delete_vote('John', 'Code-Review')
```

Parameters

- **account** –
- **label** –

- **input** – the DeleteVoteInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-changes.html#delete-vote-input

Returns

fix(*input_=None*)

Performs consistency checks on the change as with GET /check, and additionally fixes any problems that can be fixed automatically. The returned field values reflect any fixes. Some fixes have options controlling their behavior, which can be set in the FixInput entity body. Only the change owner, a project owner, or an administrator may fix changes.

```
input_ = {
    "delete_patch_set_if_commit_missing": "true",
    "expect_merged_as": "something"
}

change = client.changes.get('Project~stable~
↪I10394472cbd17dd12454f229e4f6de00b143a444')
result = change.fix()
# or
result = change.fix(input_)
```

Parameters

input – the FixInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-changes.html#fix-input

Returns

get_assignee()

Retrieves the account of the user assigned to a change.

Returns

get_attention_set()

Returns all users that are currently in the attention set. support this method since v3.3.0

Returns

get_detail(*options=None*)

retrieve a change with labels, detailed labels, detailed accounts, reviewer updates, and messages.

Parameters

options – List of options to fetch additional data about a change

Returns

get_edit()

Retrieves a change edit details. As response an EditInfo entity is returned that describes the change edit, or 204 No Content when change edit doesn't exist for this change.

Returns

get_hashtags()

Gets the hashtags associated with a change.

Returns

`get_include_in`()

Retrieves the branches and tags in which a change is included.

Returns

`get_meta_diff`(*old=None*, *meta=None*)

Retrieves the difference between two historical states of a change by specifying the and the parameters. old=SHA-1,meta=SHA-1. If the parameter is not provided, the parent of the SHA-1 is used. If the parameter is not provided, the current state of the change is used. If neither are provided, the difference between the current state of the change and its previous state is returned.

Parameters

- `old` –
- `meta` –

Returns

`get_past_assignees`()

Returns a list of every user ever assigned to a change, in the order in which they were first assigned.

Returns

`get_pure_revert`(*commit*)

Check if the given change is a pure revert of the change it references in revertOf.

Parameters
`commit` – commit id

Returns

`get_revision`(*revision_id='current'*)

Get one revision by revision SHA or integer number.

Parameters
`revision_id` – Optional ID. If not specified, the current revision will be retrieved. It supports SHA IDs and integer numbers from -X to +X, where X is the current (latest) revision. Zero means current revision. -N means the current revision number X minus N, so if the current revision is 50, and -1 is given, the revision 49 will be retrieved.

Returns

`get_topic`()

Retrieves the topic of a change.

Getter
Retrieves the topic of a change.

Setter
Sets the topic of a change.

Deleter
Deletes the topic of a change.

Returns

`ignore`()

Marks a change as ignored. The change will not be shown in the incoming reviews' dashboard, and email notifications will be suppressed. Ignoring a change does not cause the change's "updated" timestamp to be modified, and the owner is not notified.

Returns

index()

Adds or updates the change in the secondary index.

Returns

list_comments()

Lists the published comments of all revisions of the change.

Returns

list_drafts()

Lists the draft comments of all revisions of the change that belong to the calling user.

Returns

list_robot_comments()

Lists the robot comments of all revisions of the change.

Returns

list_submitted_together_changes()

Computes list of all changes which are submitted when Submit is called for this change, including the current change itself.

list_votes(*account*)

Lists the votes for a specific reviewer of the change.

Parameters

account – account id or username

Returns

mark_as_reviewed()

Marks a change as reviewed.

Returns

mark_as_unreviewed()

Marks a change as unreviewed.

Returns

mark_private(*input_*)

Marks the change to be private. Only open changes can be marked private. Changes may only be marked private by the owner or site administrators.

```
input_ = {
    "message": "After this security fix has been released we can make it public␣
↪now."
}
change = client.changes.get('Project~stable~
↪I10394472cbd17dd12454f229e4f6de00b143a444')
change.mark_private(input_)
```

Parameters

input – the PrivateInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-changes.html#private-input

Returns

property messages

move(*input_*)

Move a change. If the change cannot be moved because the change state doesn't allow moving the change, the response is '409 Conflict' and the error message is contained in the response body.

```
input_ = {
    "destination_branch" : "release-branch"
}

change = client.changes.get('Project~stable~
↪I10394472cbd17dd12454f229e4f6de00b143a444')
result = change.move(input_)
```

Parameters

input – the MoveInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-changes.html#move-input

Returns

rebase(*input_*)

Rebase a change. If the change cannot be rebased, e.g. due to conflicts, the response is '409 Conflict' and the error message is contained in the response body.

```
input_ = {
    "base" : "1234",
}

change = client.changes.get('Project~stable~
↪I10394472cbd17dd12454f229e4f6de00b143a444')
result = change.rebase(input_)
```

Parameters

input – the RebaseInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-changes.html#rebase-input

Returns

remove_from_attention_set(*id_*, *input_=None*)

Deletes a single user from the attention set of a change. support this method since v3.3.0

A user can only be removed from the attention set. if they are currently in the attention set. Otherwise, the request is silently ignored.

```
input_ = {
    "reason": "reason"
}
change = client.changes.get('Project~stable~
↪I10394472cbd17dd12454f229e4f6de00b143a444')
change.remove_from_attention_set('kevin.shi', input_)
# or
change.remove_from_attention_set('kevin.shi')
```

Parameters

- **id** – account id
- **input** – the AttentionSetInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-changes.html#attention-set-input

Returns

restore()

Restores a change. If the change cannot be restored because the change state doesn't allow restoring the change, the response is "409 Conflict" and the error message is contained in the response body.

Returns

revert(*input_=None*)

Reverts a change. The request body does not need to include a RevertInput entity if no review comment is added.

If the user doesn't have revert permission on the change or upload permission on the destination branch, the response is '403 Forbidden', and the error message is contained in the response body.

If the change cannot be reverted because the change state doesn't allow reverting the change, the response is 409 Conflict and the error message is contained in the response body.

```
input_ = {
    "message" : "Message to be added as review comment to the change when␣
↪reverting the
    change."
}

change = client.changes.get('Project~stable~
↪I10394472cbd17dd12454f229e4f6de00b143a444')
result = change.revert()
# or
result = change.revert(input_)
```

Parameters

input – the RevertInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-changes.html#revert-input

Returns

revert_submission()

Creates open revert changes for all of the changes of a certain submission.

If the user doesn't have revert permission on the change or upload permission on the destination, the response is '403 Forbidden', and the error message is contained in the response body.

If the change cannot be reverted because the change state doesn't allow reverting the change the response is '409 Conflict', and the error message is contained in the response body.

Returns

property reviewers

set_assignee(*input_*)

Sets the assignee of a change.

```
input_ = {
    "assignee": "jhon.doe"
}

change = client.changes.get('Project~stable~
↪I10394472cbd17dd12454f229e4f6de00b143a444')
result = change.set_assignee(input_)
```

Parameters

input – the AssigneeInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-changes.html#assignee-input

Returns

set_commit_message(*input_*)

Creates a new patch set with a new commit message.

```
input_ = {
    "message": "New Commit message \n\nChange-Id:␣
↪I10394472cbd17dd12454f22b143a444\n"
}

change = client.changes.get('Project~stable~
↪I10394472cbd17dd12454f229e4f6de00b143a444')
result = change.set_commit_message(input_)
```

Parameters

input – the CommitMessageInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-changes.html#commit-message-input

Returns

set_hashtags(*input_*)

Adds and/or removes hashtags from a change.

```
input_ = {
    "add" : [
        "hashtag3"
    ],
    "remove" : [
        "hashtag2"
    ]
}
change = client.changes.get('Project~stable~
↪I10394472cbd17dd12454f229e4f6de00b143a444')
result = change.set_hashtags(input_)
```

Parameters

input – the HashtagsInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-changes.html#hashtags-input

Returns

set_ready_for_review(*input_*)

Marks the change as ready for review (set WIP property to false). Changes may only be marked ready by the owner, project owners or site administrators. Marking a change ready for review also adds all of the reviewers of the change to the attention set.

```
input_ = {
    'message': 'Refactoring is done.'
}

change = client.changes.get('Project~stable~
↪I10394472cbd17dd12454f229e4f6de00b143a444')
change.set_ready_for_review(input_)
```

Parameters

input – the WorkInProgressInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-changes.html#work-in-progress-input

Returns

set_topic(*topic*)

Sets the topic of a change.

Parameters

topic – The new topic

Returns

set_work_in_progress(*input_=None*)

Marks the change as not ready for review yet. Changes may only be marked not ready by the owner, project owners or site administrators. Marking a change work in progress also removes all users from the attention set.

The request body does not need to include a WorkInProgressInput entity if no review comment is added.

```
input_ = {
    "message": "Refactoring needs to be done before we can proceed here."
}

change = client.changes.get('Project~stable~
↪I10394472cbd17dd12454f229e4f6de00b143a444')
result = change.set_work_in_progress(input_)
# or
result = change.set_work_in_progress()
```

Parameters

input – the WorkInProgressInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-changes.html#work-in-progress-input

Returns

submit(*input_=None*)

Submits a change. Submitting a change also removes all users from the attention set.

If the change cannot be submitted because the submit rule doesn't allow submitting the change, the response is 409 Conflict and the error message is contained in the response body.

```
input_ = {
    "on_behalf_of": 1001439
}

change = client.changes.get('Project~stable~
↪I10394472cbd17dd12454f229e4f6de00b143a444')
result = change.submit(input_)
```

Parameters
input – the SubmitInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-changes.html#submit-input

Returns

unignore()

Un-marks a change as ignored.

Returns

unmark_private(*input_=None*)

Marks the change to be non-private. Note users can only unmark own private changes. If the change was already not private, the response is '409 Conflict'.

```
input_ = {
    "message": "This is a security fix that must not be public."
}
change = client.changes.get('Project~stable~
↪I10394472cbd17dd12454f229e4f6de00b143a444')
change.unmark_private(input_)
# or
change.unmark_private()
```

Parameters
input – the PrivateInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-changes.html#private-input

Returns

gerrit.changes.changes module

class gerrit.changes.changes.**GerritChanges**(*gerrit*)

Bases: object

create(*input_*)

create a change

```
input_ = {
    "project": "myProject",
    "subject": "Let's support 100% Gerrit workflow direct in browser",
    "branch": "stable",
    "topic": "create-change-in-browser",
    "status": "NEW"
```

(continues on next page)

(continued from previous page)

```
}
result = client.changes.create(input_)
```

Parameters
input – the ChangeInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-changes.html#change-input

Returns

delete(*id_*)

Deletes a change.

Parameters
id – change id

Returns

get(*id_*)

Retrieves a change.

Parameters
id – change id

Returns

search(*query: str*, *options=None*, *limit: int = 25*, *skip: int = 0*)

Queries changes visible to the caller.

```
query = "is:open+owner:self+is:mergeable"
result = client.changes.search(query=query, options=["LABELS"])
```

Parameters

- **query** – Query string, it can contain multiple search operators concatenated by '+' character
- **options** – List of options to fetch additional data about changes
- **limit** – Int value that allows to limit the number of changes to be included in the output results
- **skip** – Int value that allows to skip the given number of changes from the beginning of the list

Returns

gerrit.changes.edit module

class gerrit.changes.edit.**GerritChangeEdit**(*change: str*, *gerrit*)

Bases: object

change_commit_message(*input_*)

Modify commit message.

```
input_ = {
    "message": "New commit message\n\n
    Change-Id: I10394472cbd17dd12454f229e4f6de00b143a444"
}

change = client.changes.get('Project~stable~
↪I10394472cbd17dd12454f229e4f6de00b143a444')
edit = change.get_edit()
edit.change_commit_message(input_)
```

Parameters
input – the ChangeEditMessageInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-changes.html#change-edit-message-input

Returns

delete()
Deletes change edit.

Returns

delete_file(*file*)
Deletes a file from a change edit.

Parameters
file – Path to file to delete.

Returns

get_change_file_content(*file*)
Retrieves content of a file from a change edit. The content of the file is returned as text encoded inside base64.

Parameters
file – the file path

Returns

get_commit_message()
Retrieves commit message from change edit. The commit message is returned as base64 encoded string.

Returns

get_file_meta_data(*file*)
Retrieves meta data of a file from a change edit.

Parameters
file – the file path

Returns

publish(*input_*)
Promotes change edit to a regular patch set.

```
input_ = {
    "notify": "NONE"
}
```

(continues on next page)

(continued from previous page)

```
change = client.changes.get('Project~stable~
↪I10394472cbd17dd12454f229e4f6de00b143a444')
edit = change.get_edit()
edit.publish(input_)
```

Parameters

input – the PublishChangeEditInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-changes.html#publish-change-edit-input

Returns

put_change_file_content(*file*, *file_content*)

Put content of a file to a change edit.

Parameters

- **file** – the file path
- **file_content** – the content of the file need to change

Returns

rebase()

Rebase change edit on top of the latest patch set. When change was rebased on top of the latest patch set, response '204 No Content' is returned. When change edit is already based on top of the latest patch set, the response '409 Conflict' is returned.

Returns

rename_file(*old_path*, *new_path*)

rename file

Parameters

- **old_path** – Old path to file to rename.
- **new_path** – New path to file to rename.

Returns

restore_file_content(*file*)

restores file content

Parameters

file – Path to file to restore.

Returns

gerrit.changes.messages module

class gerrit.changes.messages.**GerritChangeMessage**(*id: str*, *change: str*, *gerrit*)

Bases: *GerritBase*

delete(*input_=None*)

Deletes a change message. Note that only users with the Administrate Server global capability are permitted to delete a change message.

```
input_ = {
    "reason": "spam"
}
change = client.changes.get('Project~stable~
↪I10394472cbd17dd12454f229e4f6de00b143a444')
message = change.messages.get("babf4c5dd53d7a11080696efa78830d0a07762e6")
result = message.delete(input_)
# or
result = message.delete()
```

Parameters
input – the DeleteChangeMessageInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-changes.html#delete-change-message-input

Returns

class gerrit.changes.messages.**GerritChangeMessages**(*change*, *gerrit*)

Bases: object

get(*id_*)

Retrieves a change message including detailed account information.

Parameters
id – change message id

Returns

list()

Lists all the messages of a change including detailed account information.

Returns

gerrit.changes.reviewers module

class gerrit.changes.reviewers.**GerritChangeReviewer**(*account: str*, *change: str*, *gerrit*)

Bases: *GerritBase*

delete(*input_=None*)

Deletes a reviewer from a change. Deleting a reviewer also removes that user from the attention set.

```
input_ = {
    "notify": "NONE"
}

change = client.changes.get('Project~stable~
↪I10394472cbd17dd12454f229e4f6de00b143a444')
reviewer = change.reviewers.get('john.doe')
reviewer.delete(input_)
# or
reviewer.delete()
```

Parameters
input – the DeleteReviewerInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-changes.html#delete-reviewer-input

Returns

delete_vote(*label*, *input_=None*)

Deletes a single vote from a change. Note, that even when the last vote of a reviewer is removed the reviewer itself is still listed on the change.

```
input_ = {
    "notify": "NONE"
}

change = client.changes.get('Project~stable~
↪I10394472cbd17dd12454f229e4f6de00b143a444')
reviewer = change.reviewers.get('john.doe')
reviewer.delete_vote('Code-Review', input_)
# or
reviewer.delete_vote('Code-Review')
```

Parameters

- **label** –
- **input** – the DeleteVoteInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-changes.html#delete-vote-input

Returns

list_votes()

Lists the votes for a specific reviewer of the change.

Returns

class gerrit.changes.reviewers.**GerritChangeReviewers**(*change*, *gerrit*)

Bases: object

add(*input_*)

Adds one user or all members of one group as reviewer to the change.

Users can be moved from reviewer to CC and vice versa. This means if a user is added as CC that is already a reviewer on the change, the reviewer state of that user is updated to CC. If a user that is already a CC on the change is added as reviewer, the reviewer state of that user is updated to reviewer.

Adding a new reviewer also adds that reviewer to the attention set, unless the change is work in progress. Also, moving a reviewer to CC removes that user from the attention set.

```
input_ = {
    "reviewer": "john.doe"
}

change = client.changes.get('Project~stable~
↪I10394472cbd17dd12454f229e4f6de00b143a444')
new_reviewer = change.reviewers.add(input_)
```

Parameters

input – the ReviewerInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-changes.html#reviewer-input

Returns

get(*account*)

Retrieves a reviewer of a change.

Parameters
account – _account_id, name, username or email

Returns

list()

Lists the reviewers of a change.

Returns

Module contents

gerrit.config package

Submodules

gerrit.config.caches module

class gerrit.config.caches.**Cache**(*name: str*, *gerrit*)

Bases: object

flush()

Flushes a cache.

Returns

class gerrit.config.caches.**Caches**(*gerrit*)

Bases: object

flush(*name*)

Flushes a cache.

Parameters
name – cache name

Returns

get(*name*)

Retrieves information about a cache.

Parameters
name – cache name

Returns

list()

Lists the caches of the server. Caches defined by plugins are included.

Returns

operation(*input_*)

Cache Operations

```
input_ = {
    "operation": "FLUSH_ALL"
}
gerrit.config.caches.operation(input_)
```

Parameters
input – the CacheOperationInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-config.html#cache-operation-input

Returns

gerrit.config.config module

class gerrit.config.config.**GerritConfig**(*gerrit*)

Bases: object

property caches

check_consistency(*input_*)

Runs consistency checks and returns detected problems.

```
input_ = {
    "check_accounts": {},
    "check_account_external_ids": {}
}
result = client.config.check_consistency(input_)
```

Parameters
input – the ConsistencyCheckInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-config.html#consistency-check-input

Returns

confirm_email(*input_*)

Confirms that the user owns an email address. If the token is invalid or if it's the token of another user the request fails and the response is '422 Unprocessable Entity'.

```
input_ = {
    "token": "Enim+QNbAo6TV8Hur8WwoUypI6apG7qBPvF+bw==
↪$MTAwMDAwNDp0ZXN0QHRlc3QuZGU="
}
result = client.config.confirm_email(input_)
```

Parameters
input – the EmailConfirmationInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-config.html#email-confirmation-input

Returns

get_default_diff_preferences()

Returns the default diff preferences for the server.

Returns

get_default_edit_preferences()

Returns the default edit preferences for the server.

Returns

get_default_user_preferences()

Returns the default user preferences for the server.

Returns

get_server_info()

get the information about the Gerrit server configuration.

Returns

get_summary(*option=None*)

Retrieves a summary of the current server state.

Parameters
option – query option.such as jvm or gc

Returns

get_top_menus()

Returns the list of additional top menu entries.

Returns

get_version()

get the version of the Gerrit server.

Returns

index_changes(*input_*)

Index a set of changes

```
input_ = {changes: ["foo~101", "bar~202"]}
gerrit.config.index_changes(input_)
```

Parameters
input – the IndexChangesInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-config.html#index-changes-input

Returns

list_capabilities()

Lists the capabilities that are available in the system. There are two kinds of capabilities: core and plugin-owned capabilities.

Returns

reload_config()

Reloads the gerrit.config configuration.

Returns

set_default_diff_preferences(*input_*)

Sets the default diff preferences for the server.

```
input_ = {
    "context": 10,
    "tab_size": 8,
    "line_length": 80,
    "cursor_blink_rate": 0,
    "intraline_difference": true,
    "show_line_endings": true,
    "show_tabs": true,
    "show_whitespace_errors": true,
    "syntax_highlighting": true,
    "auto_hide_diff_table_header": true,
    "theme": "DEFAULT",
    "ignore_whitespace": "IGNORE_NONE"
}
result = client.config.set_default_diff_preferences(input_)
```

Parameters

input – the DiffPreferencesInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-accounts.html#diff-preferences-input

Returns

set_default_edit_preferences(*input_*)

Sets the default edit preferences for the server.

```
input_ = {
    "tab_size": 8,
    "line_length": 80,
    "indent_unit": 2,
    "cursor_blink_rate": 0,
    "show_tabs": true,
    "syntax_highlighting": true,
    "match_brackets": true,
    "auto_close_brackets": true,
    "theme": "DEFAULT",
    "key_map_type": "DEFAULT"
}
result = client.config.set_default_edit_preferences(input_)
```

Parameters

input – the EditPreferencesInfo entity, https://gerrit-review.googlesource.com/Documentation/rest-api-accounts.html#edit-preferences-input

Returns

set_default_user_preferences(*input_*)

Sets the default user preferences for the server.

```
input_ = {
    "changes_per_page": 50
}
result = client.config.set_default_user_preferences(input_)
```

Parameters
input – the PreferencesInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-accounts.html#preferences-input

Returns

property tasks

gerrit.config.tasks module

class gerrit.config.tasks.Task(*task_id: str*, *gerrit*)

Bases: object

delete()

Kills a task from the background work queue that the Gerrit daemon is currently performing, or will perform in the near future.

Returns

class gerrit.config.tasks.Tasks(*gerrit*)

Bases: object

delete(*id_*)

Kills a task from the background work queue that the Gerrit daemon is currently performing, or will perform in the near future.

Parameters
id – task id

Returns

get(*id_*)

Retrieves a task from the background work queue that the Gerrit daemon is currently performing, or will perform in the near future.

Parameters
id – task id

Returns

list()

Lists the tasks from the background work queues that the Gerrit daemon is currently performing, or will perform in the near future.

Returns

Module contents

gerrit.groups package

Submodules

gerrit.groups.group module

class gerrit.groups.group.**GerritGroup**(*group_id: int*, *gerrit*)

Bases: *GerritBase*

delete_description()

Deletes the description of a Gerrit internal group. This endpoint is only allowed for Gerrit internal groups; attempting to call on a non-internal group will return 405 Method Not Allowed.

Returns

get_audit_log()

Gets the audit log of a Gerrit internal group. This endpoint is only allowed for Gerrit internal groups; attempting to call on a non-internal group will return 405 Method Not Allowed.

Returns

get_description()

Retrieves the description of a group.

Returns

get_detail()

Retrieves a group with the direct members and the directly included groups.

Returns

get_name()

Retrieves the name of a group.

Returns

get_options()

Retrieves the options of a group.

Returns

get_owner()

Retrieves the owner group of a Gerrit internal group.

Returns

As response a GroupInfo entity is returned that describes the owner group.

index()

Adds or updates the internal group in the secondary index.

Returns

property members

set_description(*input_*)

Sets the description of a Gerrit internal group. This endpoint is only allowed for Gerrit internal groups; attempting to call on a non-internal group will return 405 Method Not Allowed.

```
input_ = {
    "description": "The committers of MyProject."
}
group = client.groups.get('0017af503a22f7b3fa6ce2cd3b551734d90701b4')
result = group.set_description(input_)
```

Parameters

input –

Returns

set_name(*input_*)

Renames a Gerrit internal group. This endpoint is only allowed for Gerrit internal groups; attempting to call on a non-internal group will return 405 Method Not Allowed.

```
input_ = {
    "name": "My Project Committers"
}

group = client.groups.get('0017af503a22f7b3fa6ce2cd3b551734d90701b4')
result = group.set_name(input_)
```

Parameters
input –

Returns

set_options(*input_*)

Sets the options of a Gerrit internal group. This endpoint is only allowed for Gerrit internal groups; attempting to call on a non-internal group will return 405 Method Not Allowed.

```
input_ = {
    "visible_to_all": True
}
group = client.groups.get('0017af503a22f7b3fa6ce2cd3b551734d90701b4')
result = group.set_options(input_)
```

Parameters
input – the GroupOptionsInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-groups.html#group-options-input

Returns

set_owner(*input_*)

Sets the owner group of a Gerrit internal group. This endpoint is only allowed for Gerrit internal groups; attempting to call on a non-internal group will return 405 Method Not Allowed.

```
input_ = {
    "owner": "6a1e70e1a88782771a91808c8af9bbb7a9871389"
}
group = client.groups.get('0017af503a22f7b3fa6ce2cd3b551734d90701b4')
result = group.set_owner(input_)
```

Parameters
input – As response a GroupInfo entity is returned that describes the new owner

group. :return:

property subgroup

gerrit.groups.groups module

class gerrit.groups.groups.**GerritGroups**(*gerrit*)

Bases: object

create(*name*, *input_*)

Creates a new Gerrit internal group.

```
input_ = {
    "description": "contains all committers for MyProject2",
    "visible_to_all": 'true',
    "owner": "Administrators",
    "owner_id": "af01a8cb8cbd8ee7be072b98b1ee882867c0cf06"
}
new_group = client.groups.create('My-Project2-Committers', input_)
```

Parameters

- **name** – group name
- **input** – the GroupInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-groups.html#group-input

Returns

get(*id_*)

Retrieves a group.

Parameters

- **id** – group id, or group_id, or group name
- **detailed** –

Returns

list(*pattern_dispatcher=None*, *options=None*, *limit: int = 25*, *skip: int = 0*)

Lists the groups accessible by the caller.

Parameters

- **pattern_dispatcher** – Dict of pattern type with respective pattern value: {('match'|'regex') : value}
- **options** – Additional fields can be obtained by adding o parameters, each option requires more lookups and slows down the query response time to the client so they are generally disabled by default. Optional fields are:

  INCLUDES: include list of direct subgroups. MEMBERS: include list of direct group members.
- **limit** – Int value that allows to limit the number of groups to be included in the output results
- **skip** – Int value that allows to skip the given number of groups from the beginning of the list

Returns

search(*query*, *options=None*, *limit: int = 25*, *skip: int = 0*)

Query Groups

Parameters

- **query** –
- **options** – Additional fields can be obtained by adding o parameters, each option requires more lookups and slows down the query response time to the client so they are generally disabled by default. Optional fields are:

  INCLUDES: include list of direct subgroups. MEMBERS: include list of direct group members.
- **limit** – Int value that allows to limit the number of groups to be included in the output results
- **skip** – Int value that allows to skip the given number of groups from the beginning of the list

Returns

gerrit.groups.members module

class gerrit.groups.members.**GerritGroupMembers**(*group_id*, *gerrit*)

Bases: object

add(*account*)

Adds a user as member to a Gerrit internal group. This endpoint is only allowed for Gerrit internal groups; attempting to call on a non-internal group will return 405 Method Not Allowed.

Parameters

account – account username or id

Returns

get(*account*)

Retrieves a group member. This endpoint is only allowed for Gerrit internal groups; attempting to call on a non-internal group will return 405 Method Not Allowed.

Parameters

account – account username or id

Returns

list()

Lists the direct members of a Gerrit internal group. This endpoint is only allowed for Gerrit internal groups; attempting to call on a non-internal group will return 405 Method Not Allowed.

Returns

remove(*account*)

Removes a user from a Gerrit internal group. This endpoint is only allowed for Gerrit internal groups; attempting to call on a non-internal group will return 405 Method Not Allowed.

Parameters

account – account username or id

Returns

gerrit.groups.subgroups module

class gerrit.groups.subgroups.**GerritGroupSubGroups**(*group_id*, *gerrit*)

Bases: object

add(*subgroup*)

Adds an internal or external group as subgroup to a Gerrit internal group. This endpoint is only allowed for Gerrit internal groups; attempting to call on a non-internal group will return 405 Method Not Allowed.

Parameters
subgroup – subgroup id or name

Returns

get(*subgroup*)

Retrieves a subgroup. This endpoint is only allowed for Gerrit internal groups; attempting to call on a non-internal group will return 405 Method Not Allowed.

Parameters
subgroup – subgroup id or name

Returns

list()

Lists the direct subgroups of a group. This endpoint is only allowed for Gerrit internal groups; attempting to call on a non-internal group will return 405 Method Not Allowed.

Returns

remove(*subgroup*)

Removes a subgroup from a Gerrit internal group. This endpoint is only allowed for Gerrit internal groups; attempting to call on a non-internal group will return 405 Method Not Allowed.

Parameters
subgroup – subgroup id or name

Returns

Module contents

gerrit.plugins package

Submodules

gerrit.plugins.plugins module

class gerrit.plugins.plugins.**GerritPlugin**(*id_: str*, *gerrit*)

Bases: object

disable()

Disables a plugin on the Gerrit server.

Returns

enable()

Enables a plugin on the Gerrit server.

Returns

reload()

Reloads a plugin on the Gerrit server.

Returns

class gerrit.plugins.plugins.**GerritPlugins**(*gerrit*)

Bases: object

get(*id_*)

Parameters
id – plugin id

Returns

install(*id_*, *input_*)

Installs a new plugin on the Gerrit server.

```
input_ = {
    "url": "file:///gerrit/plugins/delete-project/delete-project-2.8.jar"
}

plugin = client.plugins.install(input_)
```

Parameters

- **id** – plugin id
- **input** – the PluginInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-plugins.html#plugin-input

Returns

list(*is_all: bool = False*, *limit: int = 25*, *skip: int = 0*, *pattern_dispatcher=None*)

Lists the plugins installed on the Gerrit server.

Parameters

- **is_all** – boolean value, if True then all plugins (including hidden ones) will be added to the results
- **limit** – Int value that allows to limit the number of plugins to be included in the output results
- **skip** – Int value that allows to skip the given number of plugins from the beginning of the list
- **pattern_dispatcher** – Dict of pattern type with respective pattern value: {('prefix'|'match'|'regex') : value}

Returns

Module contents

gerrit.projects package

Submodules

gerrit.projects.branches module

class gerrit.projects.branches.**GerritProjectBranch**(*name: str*, *project: str*, *gerrit*)

Bases: *GerritBase*

delete()

Delete a branch.

Returns

get_file_content(*file*, *decode=False*)

Gets the content of a file from the HEAD revision of a certain branch. The content is returned as base64 encoded string.

Parameters

- **file** – the file path
- **decode** – Decode bas64 to plain text.

Returns

get_reflog()

Gets the reflog of a certain branch.

Returns

is_mergeable(*input_*)

Gets whether the source is mergeable with the target branch.

```
input_ = {
    'source': 'testbranch',
    'strategy': 'recursive'
}
result = stable.is_mergeable(input_)
```

Parameters

input – the MergeInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-changes.html#merge-input

Returns

class gerrit.projects.branches.**GerritProjectBranches**(*project*, *gerrit*)

Bases: object

branch_prefix = 'refs/heads/'

create(*name*, *input_*)

Creates a new branch.

```
input_ = {
    'revision': '76016386a0d8ecc7b6be212424978bb45959d668'
}
project = client.projects.get('myproject')
new_branch = project.branches.create('stable', input_)
```

Parameters

- **name** – the branch name
- **input** – the BranchInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-projects.html#branch-info

Returns

delete(*name*)

Delete a branch.

Parameters
name – branch ref name

Returns

get(*name*)

get a branch by ref

Parameters
name – branch ref name

Returns

list(*pattern_dispatcher=None*, *limit: int = 25*, *skip: int = 0*)

List the branches of a project.

Parameters

- **pattern_dispatcher** – Dict of pattern type with respective pattern value: {('match'|'regex') : value}
- **limit** – Limit the number of branches to be included in the results.
- **skip** – Skip the given number of branches from the beginning of the list.

Returns

gerrit.projects.commit module

class gerrit.projects.commit.**GerritProjectCommit**(*commit: str*, *project: str*, *gerrit*)

Bases: *GerritBase*

cherry_pick(*input_*)

Cherry-picks a commit of a project to a destination branch.

```
input_ = {
    "message": "Implementing Feature X",
    "destination": "release-branch"
}
result = commit.cherry_pick(input_)
```

Parameters
input – the CherryPickInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-changes.html#cherrypick-input

Returns
the resulting cherry-picked change

get_file_content(*file*, *decode=False*)

Gets the content of a file from a certain commit.

Parameters

- **file** – the file path
- **decode** – Decode bas64 to plain text.

Returns

get_include_in()

Retrieves the branches and tags in which a change is included.

Returns

list_change_files()

Lists the files that were modified, added or deleted in a commit.

Returns

gerrit.projects.dashboards module

class gerrit.projects.dashboards.**GerritProjectDashboard**(*id: str*, *project: str*, *gerrit*)

Bases: *GerritBase*

delete()

Deletes a project dashboard.

Returns

class gerrit.projects.dashboards.**GerritProjectDashboards**(*project*, *gerrit*)

Bases: object

create(*id_*, *input_*)

Creates a project dashboard, if a project dashboard with the given dashboard ID doesn't exist yet.

```
input_ = {
    "id": "master:closed",
    "commit_message": "Define the default dashboard"
}
new_dashboard = project.dashboards.create('master:closed', input_)
```

Parameters

- **id** – the dashboard id
- **input** – the DashboardInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-projects.html#dashboard-input

Returns

delete(*id_*)

Deletes a project dashboard.

Parameters
id – the dashboard id

Returns

get(*id_*)

Retrieves a project dashboard. The dashboard can be defined on that project or be inherited from a parent project.

Parameters
id – the dashboard id

Returns

list()

List custom dashboards for a project.

Returns

gerrit.projects.labels module

class gerrit.projects.labels.**GerritProjectLabel**(*name: str*, *project: str*, *gerrit*)

Bases: *GerritBase*

delete()

Deletes the definition of a label that is defined in this project. The calling user must have write access to the refs/meta/config branch of the project.

Returns

set(*input_*)

Updates the definition of a label that is defined in this project. The calling user must have write access to the refs/meta/config branch of the project. Properties which are not set in the input entity are not modified.

```
input_ = {
    "commit_message": "Ignore self approvals for Code-Review label",
    "ignore_self_approval": true
}

project = client.projects.get("MyProject")
label = project.labels.get("foo")
result = label.set(input_)
```

Parameters
input – the LabelDefinitionInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-projects.html#label-definition-input

Returns

class gerrit.projects.labels.**GerritProjectLabels**(*project*, *gerrit*)

Bases: object

create(*name*, *input_*)

Creates a new label definition in this project. The calling user must have write access to the refs/meta/config branch of the project. If a label with this name is already defined in this project, this label definition is updated (see Set Label).

```
input_ = {
    "values": {
        " 0": "No score",
        "-1": "I would prefer this is not merged as is",
        "-2": "This shall not be merged",
        "+1": "Looks good to me, but someone else must approve",
        "+2": "Looks good to me, approved"
    },
    "commit_message": "Create Foo Label"
}
new_label = project.labels.create('foo', input_)
```

Parameters

- **name** – label name
- **input** – the LabelDefinitionInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-projects.html#label-definition-input

Returns

delete(*name*)

Deletes the definition of a label that is defined in this project. The calling user must have write access to the refs/meta/config branch of the project.

Parameters
name – label name

Returns

get(*name*)

Retrieves the definition of a label that is defined in this project. The calling user must have read access to the refs/meta/config branch of the project.

Parameters
name – label name

Returns

list()

Lists the labels that are defined in this project.

Returns

gerrit.projects.project module

class gerrit.projects.project.**GerritProject**(*project_id: str*, *gerrit*)

Bases: *GerritBase*

ban_commits(*input_*)

Marks commits as banned for the project.

```
input_ = {
    "commits": [
      "a8a477effbbf3b44169bb9a1d3a334cbbd9aa96",
      "cf5b56541f84b8b57e16810b18daca9c3adc377b"
    ],
    "reason": "Violates IP"
}
project = client.projects.get('myproject')
result = project.ban_commits(input_)
```

Parameters

input – the BanInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-projects.html#ban-input

Returns

property branches

List the branches of a project. except the refs/meta/config

Returns

check_access(*options*)

runs access checks for other users.

Parameters

options –

Check Access Options

- Account(account): The account for which to check access. Mandatory.
- Permission(perm): The ref permission for which to check access. If not specified, read access to at least branch is checked.
- Ref(ref): The branch for which to check access. This must be given if perm is

specified.

Returns

check_consistency(*input_*)

Performs consistency checks on the project.

```
input_ = {
    "auto_closeable_changes_check": {
        "fix": 'true',
        "branch": "refs/heads/master",
        "max_commits": 100
    }
```

(continues on next page)

(continued from previous page)

```
}

project = client.projects.get('myproject')
result = project.check_consistency(input_)
```

Parameters

input – the CheckProjectInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-projects.html#check-project-input

Returns

property child_projects

List the direct child projects of a project.

Returns

create_access_rights_change(*input_*)

Sets access rights for the project using the diff schema provided by ProjectAccessInput This takes the same input as Update Access Rights, but creates a pending change for review. Like Create Change, it returns a ChangeInfo entity describing the resulting change. https://gerrit-review.googlesource.com/Documentation/rest-api-projects.html#create-access-change

Parameters

input – the ProjectAccessInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-projects.html#project-access-input

Returns

create_change(*input_*)

Create Change for review. This endpoint is functionally equivalent to create change in the change API, but it has the project name in the URL, which is easier to route in sharded deployments. support this method since v3.3.0

```
input_ = {
    "subject": "Let's support 100% Gerrit workflow direct in browser",
    "branch": "stable",
    "topic": "create-change-in-browser",
    "status": "NEW"
}

project = client.projects.get('myproject')
result = project.create_change(input_)
```

Parameters

input –

Returns

property dashboards

gerrit dashboards operations

Returns

delete()

Delete the project, requires delete-project plugin

Returns

delete_description()

Deletes the description of a project.

Returns

get_access_rights()

Lists the access rights for a single project.

Returns

get_commit(*commit*)

Retrieves a commit of a project.

Returns

get_config()

Gets some configuration information about a project. Note that this config info is not simply the contents of project.config; it generally contains fields that may have been inherited from parent projects.

Returns

get_description()

Retrieves the description of a project.

Returns

get_head()

Retrieves for a project the name of the branch to which HEAD points.

Returns

get_parent()

Retrieves the name of a project's parent project. For the All-Projects root project an empty string is returned.

Returns

get_statistics()

Return statistics for the repository of a project.

Returns

index(*input_*)

Adds or updates the current project (and children, if specified) in the secondary index. The indexing task is executed asynchronously in background and this command returns immediately if async is specified in the input.

```
input_ = {
    "index_children": "true"
    "async": "true"
}
project = client.projects.get('myproject')
result = project.index(input_)
```

Parameters

input – the IndexProjectInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-projects.html#index-project-input

Returns

`index_all_changes()`

Adds or updates the current project (and children, if specified) in the secondary index. The indexing task is executed asynchronously in background and this command returns immediately if async is specified in the input.

Returns

`property labels`

gerrit labels or gerrit labels operations

Returns

`run_garbage_collection(input_)`

Run the Git garbage collection for the repository of a project.

```
input_ = {
    "show_progress": "true"
}
project = client.projects.get('myproject')
result = project.run_garbage_collection(input_)
```

Parameters

input – the GCInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-projects.html#gc-input

Returns

`set_access_rights(input_)`

Sets access rights for the project using the diff schema provided by ProjectAccessInput. https://gerrit-review.googlesource.com/Documentation/rest-api-projects.html#set-access

Parameters

input – the ProjectAccessInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-projects.html#project-access-input

Returns

`set_config(input_)`

Sets the configuration of a project.

```
input_ = {
    "description": "demo project",
    "use_contributor_agreements": "FALSE",
    "use_content_merge": "INHERIT",
    "use_signed_off_by": "INHERIT",
    "create_new_change_for_all_not_in_target": "INHERIT",
    "enable_signed_push": "INHERIT",
    "require_signed_push": "INHERIT",
    "reject_implicit_merges": "INHERIT",
    "require_change_id": "TRUE",
    "max_object_size_limit": "10m",
    "submit_type": "REBASE_IF_NECESSARY",
    "state": "ACTIVE"
}
```

(continues on next page)

(continued from previous page)

```
project = client.projects.get('myproject')
result = project.set_config(input_)
```

Parameters

input – the ConfigInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-projects.html#config-info

Returns

`set_description`(*input_*)

Sets the description of a project.

```
input_ = {
    "description": "Plugin for Gerrit that handles the replication.",
    "commit_message": "Update the project description"
}
project = client.projects.get('myproject')
result = project.set_description(input_)
```

Parameters

input – the ProjectDescriptionInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-projects.html#project-description-input

Returns

`set_head`(*input_*)

Sets HEAD for a project.

```
input_ = {
    "ref": "refs/heads/stable"
}
project = client.projects.get('myproject')
result = project.set_HEAD(input_)
```

Parameters

input – The HeadInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-projects.html#head-input

Returns

`set_parent`(*input_*)

Sets the parent project for a project.

```
input_ = {
    "parent": "Public-Plugins",
    "commit_message": "Update the project parent"
}
project = client.projects.get('myproject')
result = project.set_parent(input_)
```

Parameters
input – The ProjectParentInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-projects.html#project-parent-input

Returns

property tags

List the tags of a project.

Returns

property webhooks

gerrit webhooks operations, requires webhooks plugin

Returns

gerrit.projects.projects module

class gerrit.projects.projects.**GerritProjects**(*gerrit*)

Bases: object

create(*project_name: str*, *input_: Dict[str, Any]*)

Creates a new project.

```
input_ = {
    "description": "This is a demo project.",
    "submit_type": "INHERIT",
    "owners": [
      "MyProject-Owners"
    ]
}
project = client.projects.create('MyProject', input_)
```

Parameters

- **project_name** – the name of the project
- **input** – the ProjectInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-projects.html#project-input

Returns

delete(*project_name: str*)

Delete the project, requires delete-project plugin

Parameters
project_name – project name

Returns

get(*name: str*)

Retrieves a project.

Parameters
name – the name of the project

Returns

list(*is_all: bool = False*, *limit: int = 25*, *skip: int = 0*, *pattern_dispatcher: Optional[Dict] = None*, *project_type: Optional[str] = None*, *description: bool = False*, *branch: Optional[str] = None*, *state: Optional[str] = None*) → List

Get list of all available projects accessible by the caller.

Parameters

- **is_all** – boolean value, if True then all projects (including hidden ones) will be added to the results. May not be used together with the state option.
- **limit** – Int value that allows to limit the number of projects to be included in the output results
- **skip** – Int value that allows to skip the given number of projects from the beginning of the list
- **pattern_dispatcher** – Dict of pattern type with respective pattern value: {('prefix'|'match'|'regex') : value}
- **project_type** – string value for type of projects to be fetched ('code'|'permissions'|'all')
- **description** – boolean value, if True then description will be added to the output result
- **branch** – Limit the results to the projects having the specified branch and include the sha1 of the branch in the results.
- **state** – Get all projects with the given state. May not be used together with the all

option.

Returns

search(*query: str*, *limit: int = 25*, *skip: int = 0*) → List

Queries projects visible to the caller. The query string must be provided by the query parameter. The start and limit parameters can be used to skip/limit results.

query parameter

- name:'NAME' Matches projects that have exactly the name 'NAME'.
- parent:'PARENT' Matches projects that have 'PARENT' as parent project.
- inname:'NAME' Matches projects that a name part that starts with

  'NAME' (case insensitive).

- description:'DESCRIPTION' Matches projects whose description contains 'DESCRIPTION',

using a full-text search. * state:'STATE' Matches project's state. Can be either 'active' or 'read-only'.

Parameters

- **query** –
- **limit** – Int value that allows to limit the number of accounts to be included in the output results
- **skip** – Int value that allows to skip the given number of accounts from the beginning of the list

Returns

gerrit.projects.tags module

class gerrit.projects.tags.**GerritProjectTag**(*name: str*, *project: str*, *gerrit*)

Bases: *GerritBase*

delete()

Delete a tag.

Returns

class gerrit.projects.tags.**GerritProjectTags**(*project*, *gerrit*)

Bases: object

create(*name*, *input_*)

Creates a new tag on the project.

```
input_ = {
    "message": "annotation",
    'revision': 'c83117624b5b5d8a7f86093824e2f9c1ed309d63'
}

project = client.projects.get('myproject')
new_tag = project.tags.create('1.1.8', input_)
```

Parameters

- **name** – the tag name
- **input** – the TagInput entity, https://gerrit-review.googlesource.com/Documentation/rest-api-projects.html#tag-input

Returns

delete(*name*)

Delete a tag.

Parameters
name – the tag ref

Returns

get(*name*)

get a tag by ref

Parameters
name – the tag ref

Returns

list(*pattern_dispatcher=None*, *limit: int = 25*, *skip: int = 0*)

List the tags of a project.

Parameters

- **pattern_dispatcher** – Dict of pattern type with respective pattern value: {('match'|'regex') : value}
- **limit** – Limit the number of tags to be included in the results.
- **skip** – Skip the given number of tags from the beginning of the list.

Returns

`tag_prefix = 'refs/tags/'`

gerrit.projects.webhooks module

`class gerrit.projects.webhooks.GerritProjectWebHook(name: str, project: str, gerrit)`

Bases: *GerritBase*

`delete()`

Delete a webhook for a project.

Returns

`class gerrit.projects.webhooks.GerritProjectWebHooks(project, gerrit)`

Bases: `object`

`create(name, input_)`

Create or update a webhook for a project.

```
input_ = {
    "url": "https://foo.org/gerrit-events",
    "maxTries": "3",
    "sslVerify": "true"
}

project = client.projects.get('myproject')
new_webhook = project.webhooks.create('test', input_)
```

Parameters

- **name** – the webhook name
- **input** – the RemoteInfo entity

Returns

`delete(name)`

Delete a webhook for a project.

Parameters
name – the webhook name

Returns

`get(name)`

Get information about one webhook.

Parameters
name – the webhook name

Returns

`list()`

List existing webhooks for a project.

Returns

Module contents

gerrit.utils package

Submodules

gerrit.utils.common module

gerrit.utils.common.**decode_response**(*response*)

Strip off Gerrit's magic prefix and decode a response. :returns:

Decoded JSON content as a dict, or raw text if content could not be decoded as JSON.

Raises

requests.HTTPError if the response contains an HTTP error status code.

gerrit.utils.common.**params_creator**(*tuples*, *pattern_types*, *pattern_dispatcher*)

gerrit.utils.common.**strip_trailing_slash**(*url*)

remove url's trailing slash :param url: url :return:

gerrit.utils.exceptions module

Module for custom exceptions.

Where possible we try to throw exceptions with non-generic, meaningful names.

exception gerrit.utils.exceptions.**AccountAlreadyExistsError**

Bases: *GerritAPIException*

Account already exists

exception gerrit.utils.exceptions.**AccountEmailAlreadyExistsError**

Bases: *GerritAPIException*

Account Email already exists

exception gerrit.utils.exceptions.**AccountEmailNotFoundError**

Bases: *GerritAPIException*

Account Email cannot be found

exception gerrit.utils.exceptions.**AccountNotFoundError**

Bases: *GerritAPIException*

Account cannot be found

exception gerrit.utils.exceptions.**AuthError**

Bases: *GerritAPIException*

403 Forbidden is returned if the operation is not allowed because the calling user does not have sufficient permissions.

exception gerrit.utils.exceptions.**BranchAlreadyExistsError**

Bases: *GerritAPIException*

Branch already exists

exception gerrit.utils.exceptions.**BranchNotFoundError**

Bases: *GerritAPIException*

Branch cannot be found

exception gerrit.utils.exceptions.**ChangeNotFoundError**

Bases: *GerritAPIException*

Change cannot be found

exception gerrit.utils.exceptions.**ClientError**

Bases: *GerritAPIException*

Client Error

exception gerrit.utils.exceptions.**CommitNotFoundError**

Bases: *GerritAPIException*

Commit cannot be found

exception gerrit.utils.exceptions.**ConflictError**

Bases: *GerritAPIException*

409 Conflict is returned if the request cannot be completed because the current state of the resource doesn't allow the operation.

exception gerrit.utils.exceptions.**FileContentNotFoundError**

Bases: *GerritAPIException*

File content not found

exception gerrit.utils.exceptions.**GPGKeyNotFoundError**

Bases: *GerritAPIException*

GPG key cannot be found

exception gerrit.utils.exceptions.**GerritAPIException**

Bases: Exception

Base class for all errors

exception gerrit.utils.exceptions.**GroupAlreadyExistsError**

Bases: *GerritAPIException*

Group already exists

exception gerrit.utils.exceptions.**GroupMemberAlreadyExistsError**

Bases: *GerritAPIException*

Group member already exists

exception gerrit.utils.exceptions.**GroupMemberNotFoundError**

Bases: *GerritAPIException*

Group member cannot be found

exception gerrit.utils.exceptions.**GroupNotFoundError**

Bases: *GerritAPIException*

Group cannot be found

exception gerrit.utils.exceptions.**NotAllowedError**

Bases: *GerritAPIException*

405 Method Not Allowed is returned if the resource exists but doesn't support the operation.

exception gerrit.utils.exceptions.**NotFoundError**

Bases: *GerritAPIException*

Resource cannot be found

exception gerrit.utils.exceptions.**ProjectAlreadyExistsError**

Bases: *GerritAPIException*

Project already exists

exception gerrit.utils.exceptions.**ProjectNotFoundError**

Bases: *GerritAPIException*

Project cannot be found

exception gerrit.utils.exceptions.**ReviewerAlreadyExistsError**

Bases: *GerritAPIException*

Reviewer already exists

exception gerrit.utils.exceptions.**ReviewerNotFoundError**

Bases: *GerritAPIException*

Reviewer cannot be found

exception gerrit.utils.exceptions.**SSHKeyNotFoundError**

Bases: *GerritAPIException*

SSH key cannot be found

exception gerrit.utils.exceptions.**ServerError**

Bases: *GerritAPIException*

Server Error

exception gerrit.utils.exceptions.**TagAlreadyExistsError**

Bases: *GerritAPIException*

Tag already exists

exception gerrit.utils.exceptions.**TagNotFoundError**

Bases: *GerritAPIException*

Tag cannot be found

exception gerrit.utils.exceptions.**UnauthorizedError**

Bases: *GerritAPIException*

401 Unauthorized

exception gerrit.utils.exceptions.**UnknownBranch**

Bases: KeyError, *NotFoundError*

Gerrit does not recognize the branch requested.

exception gerrit.utils.exceptions.**UnknownFile**

Bases: KeyError, *NotFoundError*

Gerrit does not recognize the revision file requested.

exception gerrit.utils.exceptions.**UnknownTag**

Bases: KeyError, *NotFoundError*

Gerrit does not recognize the tag requested.

exception gerrit.utils.exceptions.**ValidationError**

Bases: *GerritAPIException*

400 Bad Request is returned if the request is not understood by the server due to malformed syntax. E.g. 400 Bad Request is returned if JSON input is expected but the 'Content-Type' of the request

is not 'application/json' or the request body doesn't contain valid JSON.

400 Bad Request is also returned if required input fields are not set or if options are set which cannot be used together.

gerrit.utils.gerritbase module

class gerrit.utils.gerritbase.**GerritBase**(*pull=True*)

Bases: object

This appears to be the base object that all other gerrit objects are inherited from

poll()

to_dict()

Print out all the data in this object for debugging.

gerrit.utils.requester module

class gerrit.utils.requester.**Requester**(***kwargs*)

Bases: object

A class which carries out HTTP requests. You can replace this class with one of your own implementation if you require some other way to access Gerrit. This default class can handle simple authentication only.

AUTH_COOKIE = None

VALID_STATUS_CODES = [200]

static confirm_status(*res*)

check response status code :param res: :return:

delete(*url*, *headers=None*, *allow_redirects=True*, *raise_for_status: bool = True*, ***kwargs*)

Parameters

- **url** –
- **headers** –
- **allow_redirects** –
- **raise_for_status** –

- **kwargs** –

Returns

get(*url*, *params=None*, *headers=None*, *allow_redirects=True*, *stream=False*, *raise_for_status: bool = True*, ***kwargs*)

Parameters

- **url** –
- **params** –
- **headers** –
- **allow_redirects** –
- **stream** –
- **raise_for_status** –
- **kwargs** –

Returns

get_request_dict(*params=None*, *data=None*, *json=None*, *headers=None*, ***kwargs*)

Parameters

- **params** –
- **data** –
- **json** –
- **headers** –
- **kwargs** –

Returns

post(*url*, *params=None*, *data=None*, *json=None*, *files=None*, *headers=None*, *allow_redirects=True*, *raise_for_status: bool = True*, ***kwargs*)

Parameters

- **url** –
- **params** –
- **data** –
- **json** –
- **files** –
- **headers** –
- **allow_redirects** –
- **raise_for_status** –
- **kwargs** –

Returns

put(*url*, *params=None*, *data=None*, *json=None*, *files=None*, *headers=None*, *allow_redirects=True*, *raise_for_status: bool = True*, ***kwargs*)

Parameters

- **url** –
- **params** –
- **data** –
- **json** –
- **files** –
- **headers** –
- **allow_redirects** –
- **raise_for_status** –
- **kwargs** –

Returns

Module contents

Module contents

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

g

INDEX

D

E

F

G

I

K

L

M

N

O

P

T

U

V

W