helix-swarm

Release 0.7.4

Petr Belskiy

Jan 01, 2023

CONTENTS

Package supports sync and async syntax with same code base.

```
from helixswarm import SwarmAsyncClient, SwarmClient
```

CHAPTER

ONE

DOCUMENTATION

Read the Docs

Official REST API PDF

CHAPTER

TWO

INSTALLATION

There are two identical packages: `helix-swarm` and alias `helixswarm`, alias was created one year later due name confusion, to be import name and package name will the same.

```
pip3 install helixswarm
```

CHAPTER

THREE

EXAMPLES

Get review info:

```
from helixswarm import SwarmClient

client = SwarmClient('http://server/api/v9', 'user', 'password')
review = client.reviews.get_info(12345)
print(review['review']['author'])
```

Add comment to review in async way (be careful `SwarmAsyncClient` must be called inside async function):

```
import asyncio
from helixswarm import SwarmAsyncClient

async def example():
    client = SwarmAsyncClient('http://server/api/v5', 'user', 'password')
    await client.comments.add('reviews/12345', 'my awesome comment')

asyncio.run(example())
```

Update credentials handler:

```
import requests
from helixswarm import SwarmClient

def get_credentials():
    response = requests.get(...).json()
    return response['user'], response['password']

client = SwarmClient(
    'http://server/api/v9',
    'user',
    'password',
    auth_update_callback=get_credentials
)

# let's suppose credentials are valid now
review = client.reviews.get_info(12345)
print(review['review']['author'])

# now, after some time, password of user somehow changed, so our callback
# will be called, new credentials will be using for retry and future
```

(continues on next page)

(continued from previous page)

```
# here we get also correct review data instead of SwarmUnauthorizedError
# exception
review = client.reviews.get_info(12345)
print(review['review']['author'])
```

CHAPTER

FOUR

TESTING

Prerequisites: *tox*

Then just run tox, all dependencies and checks will run automatically

```
tox
```

CHAPTER

FIVE

CONTRIBUTING

Feel free to any contributions.

Mirror repositories of review board source code:

- 2022.1
- 2016.1

Latest version of code can be download from official page: https://www.perforce.com/downloads/helix-swarm

CHAPTER

SIX

OFFICIAL REST API DOCUMENTATION

API	Date	Notes
v10	October 2019	Integration with CI tools
v9	April 2018	Review append and replace changelist, 2fa, mark comment as read
v8	December 2017	Default reviewers
v7	October 2017	Groups as review participants, groups as moderators of project
v6	May 2017	Activity dashboard, archiving reviews, cleaning reviews, for voting reviews
v5	October 2016	Limiting comments to a specific review version
v4	October 2016	Private projects, file-level and line-level inline comments
v3	September 2016	Comments management
v2	May 2016	Projects, groups
v1.2	October 2015	Author filter to the list reviews endpoint
v1.1	January 201	Required reviewers
v1	July 2014	Initial

CHAPTER

SEVEN

API ENDPOINTS

7.1 Core

class helixswarm.**SwarmClient**(*url: str*, *user: str*, *password: str*, ***, *verify: bool = True*, *timeout: Optional[float] = None*, *retry: Optional[dict] = None*, *auth_update_callback: Optional[Callable[[], Tuple[str, str]]] = None*)

Swarm client class.

Parameters

- **url** (*str*) – URL of Swarm server, must include API version.
- **user** (*str*) – User name.
- **password** (*str*) – Password for user.
- **verify** (*Optional[bool]*) – Verify SSL (default: true).
- **timeout** (*Optional[float]*) – HTTP request timeout.
- **retry** (*Optional[dict]*) – Retry options to prevent failures if server restarting or temporary network problem. Disabled by default use total > 0 to enable.
  - total: `int` Total retries count.
  - **factor: `int` Sleep factor between retries (default 1)**
    {factor} * (2 ** ({number of total retries} - 1))
  - statuses: `List[int]` HTTP statues retries on. (default [])
  - **methods: `List[str]` list of HTTP methods to retry, idempotent**
    methods are used by default.

  Example:

```
retry = dict(
    total=10,
    factor=1,
    statuses=[500]
)
```

  With factor = 1

Retry number	Sleep
1	0.5 seconds
2	1.0 seconds
3	2.0 seconds
4	4.0 seconds
5	8.0 seconds
6	16.0 seconds
7	32.0 seconds
8	1.1 minutes
9	2.1 minutes
10	4.3 minutes
11	8.5 minutes
12	17.1 minutes
13	34.1 minutes
14	1.1 hours
15	2.3 hours
16	4.6 hours
17	9.1 hours
18	18.2 hours
19	36.4 hours
20	72.8 hours

- **auth_update_callback** (*Optional[Callable[[], Tuple[str, str]]]*) – Callback function which will be called on SwarmUnauthorizedError to update user and password and retry request again.

Returns
class instance.

Return type
SwarmClient

class helixswarm.**SwarmAsyncClient**(*url: str*, *user: str*, *password: str*, ***, *verify: bool = True*, *timeout: Optional[float] = None*, *retry: Optional[dict] = None*, *auth_update_callback: Optional[Callable[[], Awaitable[Tuple[str, str]]]] = None*)

Swarm async client class.

Parameters

- **url** (*str*) – URL of Swarm server, must include API version.
- **user** (*str*) – User name.
- **password** (*str*) – Password for user.
- **verify** (*Optional[bool]*) – Verify SSL (default: true).
- **timeout** (*Optional[int]*) – HTTP request timeout.
- **retry** (*Optional[dict]*) – Retry options to prevent failures if server restarting or temporary network problem. Disabled by default use total > 0 to enable.
  - total: `int` Total retries count.
  - **factor: `int` Sleep factor between retries (default 1)**
    {factor} * (2 ** ({number of total retries} - 1))

- statuses: `List[int]` HTTP statues retries on. (default [])
- **methods: `List[str]` list of HTTP methods to retry, idempotent** methods are used by default.

Example:

```
retry = dict(
    total=10,
    factor=1,
    statuses=[500]
)
```

- **auth_update_callback** (*Optional[Callable[[], Tuple[str, str]]]*) – Callback function which will be called on SwarmUnauthorizedError to update user and password and retry request again.

Returns
instance

Return type
SwarmAsyncClient

class helixswarm.swarm.**Response**(*status*, *body*)

Create new instance of Response(status, body)

property body

Alias for field number 1

property status

Alias for field number 0

class helixswarm.swarm.**Swarm**

get_version() → dict

Show server version information. This can be used to determine the currently-installed Swarm version, and also to check that Swarm's API is responding as expected.

Returns
server version.

Return type
dict

check_auth(*token: Optional[str] = None*) → dict

Checking the 2FA authentication.

Returns
check result.

Return type
dict

get_auth_methods() → dict

Returns the complete list of methods of 2FA.

Returns
auth methods.

Return type
dict

`init_auth`(*method: str*) → dict

Initiating the 2FA authentication.

Parameters
method (*str*) – the Method in which you want to use.

Returns
result response.

Return type
dict

`check_session`() → dict

Get the current effective user details.

Returns
result response.

Return type
dict

`init_session`() → dict

Create a new Swarm session using the given credentials.

Returns
result response.

Return type
dict

`destroy_session`() → dict

Destroy the current session, for instance logout.

Returns
result response.

Return type
dict

`login`(*saml: Optional[bool] = None*) → dict

Login to Swarm.

Returns
result response.

Return type
dict

`logout`() → dict

Logout of Swarm.

Returns
result response.

Return type
dict

7.2 Exceptions

exception helixswarm.exceptions.**SwarmError**

Core library exception

exception helixswarm.exceptions.**SwarmUnauthorizedError**

Raises when request return HTTP code 401 (unauthorized)

exception helixswarm.exceptions.**SwarmNotFoundError**

Raises when request return HTTP code 404 (not found)

exception helixswarm.exceptions.**SwarmCompatibleError**

Raises when trying to use new API endpoints on old API version

7.3 Activities

class helixswarm.endpoints.activities.**Activities**(*swarm*)

get(*, *change: Optional[int] = None*, *stream: Optional[str] = None*, *category: Optional[str] = None*, *after: Optional[int] = None*, *limit: Optional[int] = None*, *fields: Optional[List[str]] = None*) → dict

Retrieve the activity list.

Parameters

- **change** (*Optional[int]*) – Filter activity entries by associated changelist id. This only includes records for which there is an activity entry in Swarm.
- **stream** (*Optional[str]*) – Filter activity stream to query for entries. This can include user-initiated actions (*user-alice*), activity relating to a user's followed projects/users (*personal-alice*), review streams (*review-1234*), and project streams (*project-exampleproject*).
- **category** (*Optional[str]*) – Type of activity, examples: *change*, *comment*, *job*, *review*.
- **after** (*Optional[int]*) – An activity ID to seek to. Activity entries up to and including the specified ID are excluded from the results and do not count towards *limit*. Useful for pagination. Commonly set to the *lastSeen* property from a previous query.
- **limit** (*Optional[int]*) – Maximum number of activity entries to return. This does not guarantee that *limit* entries are returned. It does guarantee that the number of entries returned won't exceed *limit*. Server-side filtering may exclude some activity entries for permissions reasons. Default: 100.
- **fields** (*Optional[List[str]]*) – List of fields to show. Omitting this parameter or passing an empty value shows all fields.

Returns
json response.

Return type
dict

create(*, *category: str*, *user: str*, *action: str*, *target: str*, *topic: Optional[str] = None*, *description: Optional[str] = None*, *change: Optional[int] = None*, *streams: Optional[List[str]] = None*, *link: Optional[str] = None*) → dict

Retrieve the activity list.

Parameters

- **category** (*str*) – Type of activity, used for filtering activity streams. Values can include *change*, *comment*, *job*, *review*.
- **user** (*str*) – User who performed the action.
- **action** (*str*) – Action that was performed - past-tense, for example, *created*, *commented on*.
- **target** (*str*) – Target that the action was performed on, for example, *issue 1234*.
- **topic** (*Optional[str]*) – Topic for the activity entry. Topics are essentially comment thread IDs. Examples: *reviews/1234* or *jobs/job001234*.
- **description** (*Optional[str]*) – Optional description of object or activity to provide context.
- **change** (*Optional[int]*) – Optional changelist ID this activity is related to. Used to filter activity related to restricted changes.
- **streams** (*Optional[List[str]]*) – Optional array of streams to display on. This can include user-initiated actions (*user-alice*), activity relating to a user's followed projects/users (*personal-alice*), review streams (*review-1234*) and project streams (*project-exampleproject*).
- **link** (*Optional[str]*) – URL for *target*.

Returns
json response.

Return type
dict

7.4 Changes

class helixswarm.endpoints.changes.**Changes**(*swarm*)

get_affects_projects(*change: int*) → dict
Get projects, and branches, affected by a given change id (**v8+**).

Parameters
change (*int*) – Change id.

Returns
json response.

Return type
dict

get_default_reviewers(*change: int*) → dict
Get default reviewers for a given change id (**v8+**).

Parameters
change (*int*) – Change id.

Returns
json response.

Return type
dict

get_check_status(*change: int*, *category: str*) → dict

Performs checks on the change if workflow configuration requires it (**v9+**).

Parameters

- **change** (*int*) – Change id to check.
- **category** (*str*) – The type of check. Must have a value of *enforced*, *strict* or *shelve*.

Returns
json response.

Return type
dict

7.5 Comments

class helixswarm.endpoints.comments.**Comments**(*swarm*)

get(*, *after: Optional[int] = None*, *limit: Optional[int] = None*, *topic: Optional[str] = None*, *context_version: Optional[int] = None*, *ignore_archived: Optional[bool] = None*, *tasks_only: Optional[bool] = None*, *task_states: Optional[List[str]] = None*, *fields: Optional[List[str]] = None*) → dict

Get list of comments.

Parameters

- **after** (*Optional[int]*) – A comment ID to seek to. Comments up to and including the specified ID are excluded from the results and do not count towards *limit*. Useful for pagination. Commonly set to the *lastSeen* property from a previous query.
- **limit** (*Optional[int]*) – Maximum number of comments to return. This does not guarantee that *limit* comments are returned. It does guarantee that the number of comments returned won't exceed *limit*.

  Default: 100.
- **topic** (*Optional[str]*) – Only comments for given topic are returned.

  Examples: *reviews/1234*, *changes/1234*, *jobs/job001234*.
- **context_version** (*Optional[int]*) – If a *reviews/1234* topic is provided, limit returned comments to a specific version of the provided review.
- **ignore_archived** (*Optional[bool]*) – Prevents archived comments from being returned. (**v5+**)
- **task_only** (*Optional[bool]*) – Returns only comments that have been flagged as tasks. (**v5+**)
- **task_states** (*Optional[List[str]]*) – Limit the returned comments to ones that match the provided task state.

  Examples: *open*, *closed*, *verified*, or *comment*. (**v5+**)
- **fields** (*Optional[List[str]]*) – List of fields to show for each comment. Omitting this parameter or passing an empty value shows all fields.

Returns
json response.

Return type
dict

add(*topic: str*, *body: str*, ***, *silence_notification: Optional[bool] = None*, *delay_notification: Optional[bool] = None*, *task_state: Optional[str] = None*, *flags: Optional[List[str]] = None*, *context_file: Optional[str] = None*, *context_left_line: Optional[int] = None*, *context_right_line: Optional[int] = None*, *context_content: Optional[List[str]] = None*, *context_version: Optional[int] = None*) → dict

Add a comment to a topic.

Parameters

- **topic** (*str*) – Examples: *reviews/1234*, *changes/1234* or *jobs/job001234*.
- **body** (*str*) – Content of the comment, markdown is supported. https://www.perforce.com/manuals/swarm/Content/Swarm/basics.markdown.html

  Please note that sometimes message can be rendered incorrectly when markdown used then need to strip trailing spaces of message.
- **silence_notification** (*Optional[bool]*) – If true no notifications will ever be sent for this created comment.
- **delay_notification** (*Optional[bool]*) – If true notifications will be delayed.
- **task_state** (*Optional[str]*) – Task state of the comment, valid values when adding a comment are *comment* and *open*. This creates a plain comment or opens a task, respectively.
- **flags** (*Optional[List[str]]*) – Typically set to *closed* to archive a comment.
- **context_file** (*Optional[str]*) – File to comment on. Valid only for changes and reviews topics.

  Example: *//depot/main/README.txt*
- **context_left_line** (*Optional[int]*) – Left-side diff line to attach the inline comment to. Valid only for changes and reviews topics. If this is specified, *context[file]* must also be specified.
- **context_right_line** (*Optional[int]*) – Right-side diff line to attach the inline comment to. Valid only for changes and reviews topics. If this is specified, *context[file]* must also be specified.
- **context_content** (*Optional[List[str]]*) – Optionally provide content of the specified line and its four preceding lines. This is used to specify a short excerpt of context in case the lines being commented on change during the review. When not provided, Swarm makes an effort to build the content on its own - as this involves file operations, it could become slow.
- **context_version** – `int` (optional) With a `reviews` topic, this field specifies which version to attach the comment to.

Returns
json response.

Return type
dict

edit(*comment_id: int*, *body: str*, ***, *topic: Optional[str] = None*, *task_state: Optional[str] = None*, *flags: Optional[List[str]] = None*, *silence_notification: Optional[bool] = None*, *delay_notification: Optional[bool] = None*) → dict

Edit a comment.

Parameters

- **comment_id** (*int*) – ID of the comment to be edited.
- **body** (*str*) – Content of the comment.
- **topic** (*Optional[str]*) – Topic to comment on.

  Examples: *reviews/1234*, *changes/1234*, *jobs/job001234*
- **task_state** (*Optional[str]*) – Task state of the comment. Note that certain transitions, such as moving from *open* to *verified* are not possible without an intermediate step (*addressed*, in this case).

  Examples: *comment* (not a task), *open*, *addressed*, *verified*.
- **flags** (*Optional[List[str]]*) – Flags on the comment. Typically set to *closed* to archive a comment.
- **silence_notification** (*Optional[bool]*) – If set to *true* no notifications will ever be sent for this edited comment.
- **delay_notification** (*Optional[bool]*) – If set to *true* notifications will be delayed

Returns
json response.

Return type
dict

notify(*topic: str*) → dict

Sends notification for comments.

Parameters
topic (*str*) – This is going to send a single notification for any comments that were not notified immediately for the user authenticated for a given topic they are posting for.

Examples: *reviews/1234*

Returns
json response.

Return type
dict

7.6 Groups

class helixswarm.endpoints.groups.**Groups**(*swarm*)

get(*, *after: Optional[str] = None*, *limit: Optional[int] = None*, *fields: Optional[List[str]] = None*, *keywords: Optional[str] = None*) → dict

Get the complete list of groups.

Parameters

- **after** (*Optional[str]*) – A group ID to seek to. Groups prior to and including the specified ID are excluded from the results and do not count towards *limit*. Useful for pagination. Commonly set to the *lastSeen* property from a previous query.

- **limit** (*Optional[int]*) – Maximum number of groups to return. This does not guarantee that *limit* groups are returned. It does guarantee that the number of groups returned won't exceed *limit*.

  Default: 100.

- **fields** (*Optional[List[str]]*) – List of fields to show for each group. Omitting this parameter or passing an empty value shows all fields.
- **keywords** (*Optional[str]*) – Keywords to limit groups on. Only groups where the group ID, group name (if set), or description contain the specified keywords are returned.

Returns
json response.

Return type
dict

get_info(*identifier: str*, ***, *fields: Optional[List[str]]*) → dict

Retrieve information about a group.

Parameters

- **identifier** (*str*) – Group identifier.
- **fields** (*Optional[List[str]]*) – List of fields to show for each group. Omitting this parameter or passing an empty value shows all fields.

Returns
json response.

Return type
dict

create(*identifier: str*, ***, *users: Optional[List[str]] = None*, *owners: Optional[List[str]] = None*, *subgroups: Optional[List[str]] = None*, *name: Optional[str] = None*, *description: Optional[str] = None*, *email_address: Optional[str] = None*, *notify_reviews: Optional[bool] = None*, *notify_commits: Optional[bool] = None*, *use_mailing_list: Optional[bool] = None*) → dict

Create a new group.

Parameters

- **identifier** (*str*) – Group identifier.
- **users** (*Optional[List[str]]*) – An optional array of group users. **At least one of Users, Owners, or Subgroups is required.**
- **owners** (*Optional[List[str]]*) – An optional array of group owners. **At least one of Users, Owners, or Subgroups is required.**
- **subgroups** (*Optional[List[str]]*) – An optional array of group subgroups. **At least one of Users, Owners, or Subgroups is required.**
- **name** (*Optional[str]*) – An optional full name for the group.
- **description** (*Optional[str]*) – An optional group description.
- **email_address** (*Optional[str]*) – The email address for this group.
- **notify_reviews** (*Optional[bool]*) – Email members when a new review is requested.
- **notify_commits** (*Optional[bool]*) – Email members when a change is committed.
- **use_mailing_list** (*Optional[bool]*) – Whether to use the configured email address or expand individual members addresses.

Returns
json response.

Return type
dict

edit(*identifier: str*, ***, *users: Optional[List[str]] = None*, *owners: Optional[List[str]] = None*, *subgroups: Optional[List[str]] = None*, *name: Optional[str] = None*, *description: Optional[str] = None*, *email_address: Optional[str] = None*, *notify_reviews: Optional[bool] = None*, *notify_commits: Optional[bool] = None*, *use_mailing_list: Optional[bool] = None*) → dict

Change the settings of a group, only super users and group owners can perform this action.

Parameters

- **identifier** (*str*) – Group identifier.
- **users** (*Optional[List[str]]*) – An optional array of group users. **At least one of Users, Owners, or Subgroups is required.**
- **owners** (*Optional[List[str]]*) – An optional array of group owners. **At least one of Users, Owners, or Subgroups is required.**
- **subgroups** (*Optional[List[str]]*) – An optional array of group subgroups. **At least one of Users, Owners, or Subgroups is required.**
- **name** (*Optional[str]*) – An optional full name for the group.
- **description** (*Optional[str]*) – An optional group description.
- **email_address** (*Optional[str]*) – The email address for this group.
- **notify_reviews** (*Optional[bool]*) – Email members when a new review is requested.
- **notify_commits** (*Optional[bool]*) – Email members when a change is committed.
- **use_mailing_list** (*Optional[bool]*) – Whether to use the configured email address or expand individual members addresses.

Returns
json response.

Return type
dict

delete(*identifier: str*) → dict

Delete a group, only super users and group owners can perform this action.

Parameters
identifier (*str*) – Group identifier.

Returns
json response.

Return type
dict

7.7 Projects

class helixswarm.endpoints.projects.**Projects**(*swarm*)

get(*, *fields: Optional[List[str]] = None*, *workflow: Optional[str] = None*) → dict

Returns a list of projects in Swarm that are visible to the current user. Administrators will see all projects, including private ones.

Parameters

- **fields** (*Optional[List[str]]*) – List of fields to show for each group. Omitting this parameter or passing an empty value shows all fields.
- **workflow** (*Optional[str]*) – List only projects using a workflow.

Returns
json response.

Return type
dict

get_info(*identifier: str*, *, *fields: Optional[List[str]] = None*) → dict

Retrieve information about a project.

Parameters

- **identifier** (*str*) – Project identifier.
- **fields** (*Optional[List[str]]*) – List of fields to show for each project. Omitting this parameter or passing an empty value shows all fields.

Returns
json response.

Return type
dict

create(*name: str*, *members: List[str]*, *, *subgroups: Optional[List[str]] = None*, *owners: Optional[List[str]] = None*, *description: Optional[str] = None*, *is_private: Optional[bool] = None*, *deploy_config: Optional[dict] = None*, *tests_config: Optional[dict] = None*, *branches: Optional[List[dict]] = None*, *job_view: Optional[str] = None*, *notify_commits: Optional[bool] = None*, *notify_reviews: Optional[bool] = None*, *defaults: Optional[List[dict]] = None*, *retain_default_reviewers: Optional[bool] = None*, *minimum_up_votes: Optional[str] = None*) → dict

Creates a new project.

Parameters

- **name** (*str*) – Project name (is also used to generate the Project ID).
- **members** (*List[str]*) – Array of project members.
- **subgroups** (*Optional[List[str]]*) – Array of project subgroups.
- **owners** (*Optional[List[str]]*) – Array of project owners.
- **description** (*Optional[str]*) – Project description.
- **private** (*Optional[bool]*) – Private projects are visible only to: members, moderators, owners, and administrators. Default: false
- **deploy** (*Optional[dict]*) – Configuration for automated deployment.

  Example: *{'enabled': True, 'url': 'http://localhost/?change={change}'}*

- **tests** (*Optional[dict]*) – Configuration for testing, continuous integration.

  Example: *{'url': '', 'enabled': False}*

- **branches** (*Optional[List[dict]]*) – Branch definitions for this project.

  Example:

```
[
    {
    'name': 'Branch One',
    'paths': '//depot/main/TestProject/...'
    }
]
```

- **jobview** (*Optional[str]*) – Jobview for associating certain jobs with this project.

  Example: *subsystem=testproject*

- **notify_commits** (*Optional[bool]*) – Email members, moderators and followers when a change is committed.

- **notify_reviews** (*Optional[bool]*) – Email members and moderators when a new review is requested.

- **defaults** (*Optional[List[dict]]*) – Array of defaults at a project level (for example default reviewers).

  Example: *[{'reviewers': {'user2': {'required': True}}}]*

- **retain_default_reviewers** (*Optional[bool]*) – Retain the default reviewers.

- **minimum_up_votes** (*Optional[str]*) – Minimum number of up votes required.

Returns
json response.

Return type
dict

edit(*identifier: str*, ***, *name: str*, *members: List[str]*, *subgroups: Optional[List[str]] = None*, *owners: Optional[List[str]] = None*, *description: Optional[str] = None*, *is_private: Optional[bool] = None*, *deploy_config: Optional[dict] = None*, *tests_config: Optional[dict] = None*, *branches: Optional[List[dict]] = None*, *job_view: Optional[str] = None*, *notify_commits: Optional[bool] = None*, *notify_reviews: Optional[bool] = None*, *defaults: Optional[List[dict]] = None*, *retain_default_reviewers: Optional[bool] = None*, *minimum_up_votes: Optional[str] = None*) → dict

Edit a project.

Parameters

- **identifier** (*str*) – Project ID.

- **name** (*Optional[str]*) – Project name, changing the project name does not change the project ID.

- **members** (*Optional[List[str]]*) – Array of project members.

- **subgroups** (*Optional[List[str]]*) – Array of project subgroups.

- **owners** (*Optional[List[str]]*) – Array of project owners.

- **description** (*Optional[str]*) – Project description.

- **private** (*Optional[bool]*) – Private projects are visible only to: members, moderators, owners, and administrators.

  Default: false

- **deploy** (*Optional[dict]*) – Configuration for automated deployment.

  Example: *{'enabled': True, 'url': 'http://localhost/?change={change}'}*

- **tests** (*Optional[dict]*) – Configuration for testing, continuous integration.

  Example: *{'url': '', 'enabled': False}*

- **branches** (*Optional[List[dict]]*) – Branch definitions for this project.

  Example:

```
[
    {
        'name': 'Branch One',
        'paths': '//depot/main/TestProject/...'
    }
]
```

- **jobview** (*Optional[str]*) – Jobview for associating certain jobs with this project.

  Example: *subsystem=testproject*

- **notify_commits** (*Optional[bool]*) – Email members, moderators and followers when a change is committed.

- **notify_reviews** (*Optional[bool]*) – Email members and moderators when a new review is requested.

- **defaults** (*Optional[List[dict]]*) – Array of defaults at a project level (for example default reviewers).

  Example: *[{'reviewers': {'user2': {'required': True}}}]*

- **retain_default_reviewers** (*Optional[bool]*) – Retain the default reviewers.

- **minimum_up_votes** (*Optional[str]*) – Minimum number of up votes required.

Returns
json response.

Return type
dict

delete(*identifier: str*) → dict

Delete a project, mark a Swarm project as deleted. The project ID and name cannot be reused. If a project has owners set, only the owners can perform this action.

Parameters
identifier (*str*) – Project ID.

Returns
json response.

Return type
dict

7.8 Reviews

class helixswarm.endpoints.reviews.**Reviews**(*swarm*)

get(*, *after: Optional[int] = None*, *limit: Optional[int] = None*, *fields: Optional[List[str]] = None*, *authors: Optional[List[str]] = None*, *changes: Optional[List[int]] = None*, *has_reviewers: Optional[bool] = None*, *ids: Optional[List[int]] = None*, *keywords: Optional[str] = None*, *participants: Optional[List[str]] = None*, *projects: Optional[List[str]] = None*, *states: Optional[List[str]] = None*, *passes_tests: Optional[bool] = None*, *not_updated_since: Optional[str] = None*, *has_voted: Optional[str] = None*, *my_comments: Optional[bool] = None*) → dict

Get list of available reviews.

Parameters

- **after** (*Optional[int]*) – A review ID to seek to. Reviews up to and including the specified *id* are excluded from the results and do not count towards *limit*. Useful for pagination. Commonly set to the *lastSeen* property from a previous query.
- **limit** (*Optional[int]*) – Maximum number of reviews to return. This does not guarantee that *limit* reviews are returned. It does guarantee that the number of reviews returned won't exceed *limit*. Server-side filtering may exclude some reviews for permissions reasons.

  Default: *1000*
- **fields** (*Optional[List[str]*) – Fields to show, Omitting this parameter or passing an empty value shows all fields.
- **author** (*Optional[List[str]*) – One or more authors to limit reviews by. Reviews with any of the specified authors are returned. (**API v1.2+**)
- **changes** (*Optional[List[str]*) – One or more change IDs to limit reviews by. Reviews associated with any of the specified changes are returned.
- **has_reviewers** (*Optional[bool*) – Limit reviews list to those with or without reviewers.
- **ids** (*Optional[List[int]*) – One or more review IDs to fetch. Only the specified reviews are returned. This filter cannot be combined with the *limit* parameter.
- **keywords** (*Optional[str]*) – Keywords to limit reviews by. Only reviews where the description, participants list or project list contain the specified keywords are returned.
- **participants** (*Optional[List[str]*) – One or more participants to limit reviews by. Reviews with any of the specified participants are returned.
- **projects** (*Optional[List[str]*) – One or more projects to limit reviews by. Reviews affecting any of the specified projects are returned.
- **states** (*Optional[List[str]*) – One or more states to limit reviews by. Reviews in any of the specified states are returned.
- **passes_tests** (*Optional[bool]*) – Option to limit reviews by tests passing or failing.
- **not_updated_since** (*Optional[str]*) – Option to fetch unchanged reviews. Requires the date to be in the format *YYYY-mm-dd*, for example 2017-01-01. Reviews to be returned are determined by looking at the last updated date of the review.
- **has_voted** (*Optional[str]*) – Should have the value `up` or `down` to filter reviews that have been voted up or down by the current authenticated user.
- **my_comments** (*Optional[bool]*) – Filtering reviews that include comments by the current authenticated user.

Returns
json response.

Return type
dict

`get_for_dashboard`() → dict

Gets reviews for the action dashboard for the authenticated user

Returns
json response.

Return type
dict

`get_info`(*review_id: int*, *, *fields: Optional[List[str]] = None*) → dict

Retrieve information about a review.

Parameters

- **review_id** (*int*) – Review id getting information from.
- **fields** (*Optional[List[str]]*) – List of fields to show. Omitting this parameter or passing an empty value shows all fields.

Returns
json response.

Return type
dict

`get_transitions`(*review_id: int*, *, *up_voters: Optional[str] = None*) → dict

Get transitions for a review (**v9+**)

Parameters

- **review_id** – int Review id getting information from.
- **up_voters** (*Optional[str]*) – A list of users whose vote up will be assumed when determining the transitions. For example if a user has not yet voted but would be the last required vote and asked for possible transitions we would want to include 'approve'

Returns
json response.

Return type
dict

`get_latest_revision_and_change`(*review_id: int*) → Tuple[int, int]

Get latest revision and change (changelist) for a review.

Parameters
review_id (*int*) – Review id getting information from.

Returns
revision and change respectively.

Return type
Tuple[int, int]

`create`(*change: int*, *, *description: Optional[str] = None*, *reviewers: Optional[List[str]] = None*, *required_reviewers: Optional[List[str]] = None*, *reviewer_groups: Optional[List[str]] = None*) → dict

Create a review.

Parameters

- **fields** (*int*) – Change ID to create a review from.
- **description** (*Optional[str]*) – Description for the new review (defaults to change description).
- **reviewers** (*Optional[List[str]]*) – A list of reviewers for the new review.
- **required_reviewers** (*Optional[List[str]]*) – A list of required reviewers for the new review (**v1.1+**)
- **reviewer_groups** (*Optional[List[str]]*) – A list of required reviewers for the new review (**v7+**)

Returns
json response.

Return type
dict

vote(*review_id: int*, *vote: str*, ***, *version: Optional[str] = None*) → dict

Set the vote for the authenticated user to be up, down or cleared.

Parameters

- **review_id** (*int*) – Review ID.
- **vote** (*str*) – Valid votes are *up*, *down* and *clear*.
- **version** (*Optional[str]*) – Expected to be a valid review revision to vote on if supplied, ignored if the revision does not exist and the vote will apply to the latest revision.

Returns
json response.

Return type
dict

add_change(*review_id: int*, *change: int*, ***, *mode: Optional[str] = None*) → dict

Add change to a review, links the given change to the review and schedules an update.

Parameters

- **review_id** (*int*) – Review ID.
- **change** (*int*) – Change ID.
- **mode** (*str*) – The mode of operation, currently]replace` or *append*.

Returns
json response.

Return type
dict

archive(***, *not_updated_since: str*, *description: str*) → dict

Archiving the inactive reviews (**v6+**).

Parameters

- **not_updated_since** (*str*) – Updated since date. Requires the date to be in the format *YYYY-mm-dd*

  Example: *2017-01-01*

- **description** (*str*) – A description that is posted as a comment for archiving.

Returns
json response.

Return type
dict

update(*review_id: int*, *, *author: Optional[str] = None*, *description: Optional[str] = None*) → dict

Archiving the inactive reviews (**v6+**).

Parameters

- **review_id** (*int*) – Review ID.
- **author** (*Optional[str]*) – The new author for the specified review.
- **description** (*Optional[str]*) – The new description for the specified review.

Returns
json response.

Return type
dict

cleanup(*review_id: int*, *, *reopen: Optional[bool] = None*) → dict

Clean up a review for the given id (**v6+**).

Parameters

- **review_id** (*int*) – Review id getting information from.
- **reopen** (*Optional[bool]*) – Expected to be a boolean (defaulting to false). If true then an attempt will be made to reopen files into a default changelist

Returns
json response.

Return type
dict

obliterate(*review_id: int*) → dict

Obliterate a review for the given id (**v9+**).

Parameters
review_id (*int*) – Review id getting information from.

Returns
json response.

Return type
dict

7.9 Servers

class helixswarm.endpoints.servers.**Servers**(*swarm*)

get() → dict

Gets a list of servers.

Returns
json response.

Return type
dict

7.10 Users

class helixswarm.endpoints.users.**Users**(*swarm*)

get(*, *fields: Optional[List[str]] = None*, *users: Optional[List[str]] = None*, *group: Optional[str] = None*) → dict

Get list of users.

Parameters

- **fields** (*Optional[List[str]]*) – List of fields to show for each user. Omitting this parameter or passing an empty value shows all fields. Be aware the fields are case sensitive for users. You can use one of the below: User, Type, Email, Update, Access, FullName, JobView, Password, AuthMethod, Reviews.
- **users** (*Optional[List[str]]*) – List of users to display. Omitting this parameter or passing an empty value shows all users.
- **group** (*Optional[str]*) – An optional to get users from a group. Cannot be used with users parameter.

Returns
json response.

Return type
dict

unfollow_all(*name: str*) → dict

Unfollow all users and projects, admin and super users are permitted to execute unfollow all against any target user. Other users are only permitted to execute the call if they themselves are the target user

Parameters
name (*str*) – User name.

Returns
json response.

Return type
dict

7.11 Workflows

class helixswarm.endpoints.workflows.**Workflows**(*swarm*)

get(**, fields: Optional[List[str]] = None, no_cache: Optional[bool] = None*) → dict

Gets workflows.

Parameters

- **fields** (*Optional[List[str]]*) – An optional list of fields to show for each workflow. Omitting this parameter or passing an empty value shows all fields.
- **noCache** (*Optional[bool]*) – If provided and has a value of 'true' a query will always be performed and the cache of workflows is ignored. Otherwise the cache will be used if it exists.

Returns
json response.

Return type
dict

get_info(*identifier: int, *, fields: Optional[List[str]] = None*) → dict

Gets a workflow by identifier.

Parameters

- **identifier** (*int*) – Workflow id.
- **fields** (*Optional[List[str]]*) – An optional list of fields to show for each workflow. Omitting this parameter or passing an empty value shows all fields.

Returns
json response.

Return type
dict

create(*name: str, *, description: Optional[str] = None, shared: Optional[bool] = None, owners: Optional[List[str]] = None, on_submit: Optional[List[str]] = None, end_rules: Optional[List[str]] = None, auto_approve: Optional[List[str]] = None, counted_votes: Optional[str] = None*) → dict

Create a new workflow.

Parameters

- **name** (*str*) – The workflow name. Will be compared against other workflows and rejected if not unique.
- **description** (*Optional[str]*) – Description for the new workflow.
- **shared** (*Optional[bool]*) – Whether this workflow is shared for other users that do not own it. Defaults to not shared.
- **owners** (*Optional[List[str]]*) – A list owners for the workflow. Can be users or group names, prefixed with *swarm-group-*. Users and group names must exist or the workflow will be rejected.
- **on_submit** (*Optional[List[str]]*) – Data for rules when changes are submitted. Valid values for with_review are no_checking, approved, strict. Valid values for without review are: *no_checking*, *auto_create*, *reject*.

- **end_rules** (*Optional[List[str]]*) – Data for rules when changes are submitted. Valid values are: *no_checking*, *no_revision*.
- **auto_approve** (*Optional[List[str]]*) – Data for rules when changes are submitted. Valid values are: *votes*, `never`.
- **counted_votes** (*Optional[str]*) – Data for rules when counting votes up. Valid values are: *anyone*, *members*.

Returns
json response.

Return type
dict

edit(*identifier: str*, ***, *name: Optional[str] = None*, *description: Optional[str] = None*, *shared: Optional[bool] = None*, *owners: Optional[List[str]] = None*, *on_submit: Optional[dict] = None*, *end_rules: Optional[List[str]] = None*, *auto_approve: Optional[List[str]] = None*, *counted_votes: Optional[str] = None*) → dict

Edit a workflow.

Parameters

- **identifier** (*str*) – The id of the workflow being edited.
- **name** (*Optional[str]*) – The workflow name. Will be compared against other workflows and rejected if not unique.
- **description** (*Optional[str]*) – Description for the new workflow.
- **shared** (*Optional[bool]*) – Whether this workflow is shared for other users that do not own it. Defaults to not shared.
- **owners** (*Optional[List[str]*) – A list owners for the workflow. Can be users or group names, prefixed with *swarm-group-*. Users and group names must exist or the workflow will be rejected.
- **on_submit** (*Optional[dict]*) – Data for rules when changes are submitted. Valid values for with_review are no_checking, approved, strict. Valid values for without review are: *no_checking*, *auto_create*, *reject*.
- **end_rules** (*Optional[List[str]]*) – Data for rules when changes are submitted. Valid values are: *no_checking*, *no_revision*.
- **auto_approve** (*Optional[List[str]`` (optional*) – Data for rules when changes are submitted. Valid values are: *votes*, *never*.
- **counted_votes** (*Optional[str]*) – Data for rules when counting votes up. Valid values are: *anyone*, *members*.

Returns
json response.

Return type
dict

delete(*identifier: str*) → dict

Delete a workflow. This call must be authenticated and the user must have permission to edit the workflow. If the workflow is in use it cannot be deleted and an error message will be returned

Parameters
identifier (*str*) – The id of the workflow being deleted.

Returns
json response.

Return type
dict

update(*identifier: str*, *name: str*, ***, *description: Optional[str] = None*, *shared: Optional[bool] = None*, *owners: Optional[List[str]] = None*, *on_submit: Optional[dict] = None*, *end_rules: Optional[List[str]] = None*, *auto_approve: Optional[List[str]] = None*, *counted_votes: Optional[str] = None*) → dict

Update a workflow. All values should be provided in the request. If not provided any missing values are reverted to default.

Parameters

- **identifier** (*str*) – The id of the workflow being edited.
- **name** (*Optional[str]*) – The workflow name. Will be compared against other workflows and rejected if not unique.
- **description** (*Optional[str]*) – Description for the new workflow.
- **shared** (*Optional[bool]*) – Whether this workflow is shared for other users that do not own it. Defaults to not shared.
- **owners** (*Optional[List[str]]*) – A list owners for the workflow. Can be users or group names, prefixed with *swarm-group-*. Users and group names must exist or the workflow will be rejected.
- **on_submit** (*Optional[dict]*) – Data for rules when changes are submitted. Valid values for with_review are no_checking, approved, strict. Valid values for without review are: *no_checking*, *auto_create*, *reject*.
- **end_rules** (*Optional[List[str]]*) – Data for rules when changes are submitted. Valid values are: *no_checking*, *no_revision*.
- **auto_approve** (*Optional[List[str]]*) – Data for rules when changes are submitted. Valid values are: *votes*, *never*.
- **counted_votes** (*Optional[str]*) – Data for rules when counting votes up. Valid values are: *anyone*, *members*.

Returns
json response.

Return type
dict

PYTHON MODULE INDEX

h

INDEX

H

I

L

M

N

O

P

R

S

U

V

W