# qbittorrent-api

***Release 2024.1.58***

**Russell Martin**

**Jan 17, 2024**

# CONTENTS

CHAPTER

ONE

# INTRODUCTION

Python client implementation for qBittorrent Web API.

Currently supports qBittorrent v4.6.3 (Web API v2.9.3) released on January 14, 2024.

## 1.1 Features

- The entire qBittorrent Web API is implemented.
- qBittorrent version checking for an endpoint's existence/features is automatically handled.
- If the authentication cookie expires, a new one is automatically requested in line with any API call.

## 1.2 Installation

- Install via pip from PyPI:

```
python -m pip install qbittorrent-api
```

- Install a specific release (e.g. `v2022.8.34`):

```
python -m pip install qbittorrent-api==2022.8.34
```

- Install direct from `main`:

```
pip install git+https://github.com/rmartin16/qbittorrent-api.git@main#egg=qbittorrent-api
```

- Enable WebUI in qBittorrent: Tools -> Preferences -> Web UI
- If the Web API will be exposed to the Internet, follow the recommendations.

## 1.3 Getting Started

```python
import qbittorrentapi

# instantiate a Client using the appropriate WebUI configuration
conn_info = dict(
    host="localhost",
    port=8080,
    username="admin",
    password="adminadmin",
)
qbt_client = qbittorrentapi.Client(**conn_info)

# the Client will automatically acquire/maintain a logged-in state
# in line with any request. therefore, this is not strictly necessary;
# however, you may want to test the provided login credentials.
try:
    qbt_client.auth_log_in()
except qbittorrentapi.LoginFailed as e:
    print(e)

# if the Client will not be long-lived or many Clients may be created
# in a relatively short amount of time, be sure to log out:
qbt_client.auth_log_out()

# or use a context manager:
with qbittorrentapi.Client(**conn_info) as qbt_client:
    if qbt_client.torrents_add(urls="...") != "Ok.":
        raise Exception("Failed to add torrent.")

# display qBittorrent info
print(f"qBittorrent: {qbt_client.app.version}")
print(f"qBittorrent Web API: {qbt_client.app.web_api_version}")
for k, v in qbt_client.app.build_info.items():
    print(f"{k}: {v}")

# retrieve and show all torrents
for torrent in qbt_client.torrents_info():
    print(f"{torrent.hash[-6:]}: {torrent.name} ({torrent.state})")

# pause all torrents
qbt_client.torrents.pause.all()
```

## 1.4 Usage

First, the Web API endpoints are organized in to eight namespaces.

- Authentication (`auth`)
- Application (`app`)
- Log (`log`)
- Sync (`sync`)
- Transfer (`transfer`)
- Torrent Management (`torrents`)
- RSS (`rss`)
- Search (`search`)

Second, this client has two modes of interaction with the qBittorrent Web API.

Each Web API endpoint is implemented one-to-one as a method of the instantiated client.

```
import qbittorrentapi
qbt_client = qbittorrentapi.Client(host='localhost:8080', username='admin', password=
↪'adminadmin')
qbt_client.app_version()
qbt_client.rss_rules()
qbt_client.torrents_info()
qbt_client.torrents_resume(torrent_hashes='...')
# and so on
```

However, a more robust interface to the endpoints is available via each namespace. This is intended to provide a more seamless and intuitive interface to the Web API.

```
import qbittorrentapi
qbt_client = qbittorrentapi.Client(host='localhost:8080', username='admin', password=
↪'adminadmin')
# changing a preference
is_dht_enabled = qbt_client.app.preferences.dht
qbt_client.app.preferences = dict(dht=not is_dht_enabled)
# stopping all torrents
qbt_client.torrents.pause.all()
# retrieve different views of the log
qbt_client.log.main.warning()
qbt_client.log.main.normal()
```

Finally, some of the objects returned by the client support methods of their own. This is most pronounced for torrents themselves.

```
import qbittorrentapi
qbt_client = qbittorrentapi.Client(host='localhost:8080', username='admin', password=
↪'adminadmin')

for torrent in qbt_client.torrents.info.active():
    torrent.set_location(location='/home/user/torrents/')
```

(continues on next page)

(continued from previous page)

```
    torrent.reannounce()
    torrent.upload_limit = -1
```

### 1.4.1 Introduction

Python client implementation for qBittorrent Web API.

Currently supports qBittorrent v4.6.3 (Web API v2.9.3) released on January 14, 2024.

#### Features

- The entire qBittorrent Web API is implemented.
- qBittorrent version checking for an endpoint's existence/features is automatically handled.
- If the authentication cookie expires, a new one is automatically requested in line with any API call.

#### Installation

- Install via pip from PyPI:

```
python -m pip install qbittorrent-api
```

- Install a specific release (e.g. `v2022.8.34`):

```
python -m pip install qbittorrent-api==2022.8.34
```

- Install direct from `main`:

```
pip install git+https://github.com/rmartin16/qbittorrent-api.git@main#egg=qbittorrent-api
```

- Enable WebUI in qBittorrent: Tools -> Preferences -> Web UI
- If the Web API will be exposed to the Internet, follow the recommendations.

#### Getting Started

```
import qbittorrentapi

# instantiate a Client using the appropriate WebUI configuration
conn_info = dict(
    host="localhost",
    port=8080,
    username="admin",
    password="adminadmin",
)
qbt_client = qbittorrentapi.Client(**conn_info)
```

(continues on next page)

(continued from previous page)

```
# the Client will automatically acquire/maintain a logged-in state
# in line with any request. therefore, this is not strictly necessary;
# however, you may want to test the provided login credentials.
try:
    qbt_client.auth_log_in()
except qbittorrentapi.LoginFailed as e:
    print(e)

# if the Client will not be long-lived or many Clients may be created
# in a relatively short amount of time, be sure to log out:
qbt_client.auth_log_out()

# or use a context manager:
with qbittorrentapi.Client(**conn_info) as qbt_client:
    if qbt_client.torrents_add(urls="...") != "Ok.":
        raise Exception("Failed to add torrent.")

# display qBittorrent info
print(f"qBittorrent: {qbt_client.app.version}")
print(f"qBittorrent Web API: {qbt_client.app.web_api_version}")
for k, v in qbt_client.app.build_info.items():
    print(f"{k}: {v}")

# retrieve and show all torrents
for torrent in qbt_client.torrents_info():
    print(f"{torrent.hash[-6:]}: {torrent.name} ({torrent.state})")

# pause all torrents
qbt_client.torrents.pause.all()
```

### Usage

First, the Web API endpoints are organized in to eight namespaces.

- Authentication (`auth`)
- Application (`app`)
- Log (`log`)
- Sync (`sync`)
- Transfer (`transfer`)
- Torrent Management (`torrents`)
- RSS (`rss`)
- Search (`search`)

Second, this client has two modes of interaction with the qBittorrent Web API.

Each Web API endpoint is implemented one-to-one as a method of the instantiated client.

```
import qbittorrentapi
qbt_client = qbittorrentapi.Client(host='localhost:8080', username='admin', password=
```

(continues on next page)

(continued from previous page)

```
↪'adminadmin')
qbt_client.app_version()
qbt_client.rss_rules()
qbt_client.torrents_info()
qbt_client.torrents_resume(torrent_hashes='...')
# and so on
```

However, a more robust interface to the endpoints is available via each namespace. This is intended to provide a more seamless and intuitive interface to the Web API.

```
import qbittorrentapi
qbt_client = qbittorrentapi.Client(host='localhost:8080', username='admin', password=
↪'adminadmin')
# changing a preference
is_dht_enabled = qbt_client.app.preferences.dht
qbt_client.app.preferences = dict(dht=not is_dht_enabled)
# stopping all torrents
qbt_client.torrents.pause.all()
# retrieve different views of the log
qbt_client.log.main.warning()
qbt_client.log.main.normal()
```

Finally, some of the objects returned by the client support methods of their own. This is most pronounced for torrents themselves.

```
import qbittorrentapi
qbt_client = qbittorrentapi.Client(host='localhost:8080', username='admin', password=
↪'adminadmin')

for torrent in qbt_client.torrents.info.active():
    torrent.set_location(location='/home/user/torrents/')
    torrent.reannounce()
    torrent.upload_limit = -1
```

### 1.4.2 Behavior & Configuration

#### Host, Username and Password

- The authentication credentials can be provided when instantiating *Client*:

```
qbt_client = Client(host="localhost:8080", username='...', password='...')
```

- The credentials can also be specified after *Client* is created but calling *auth_log_in()* is not strictly necessary to authenticate the client; this will happen automatically for any API request.

```
qbt_client.auth_log_in(username='...', password='...')
```

- Alternatively, the credentials can be specified in environment variables:
  - QBITTORRENTAPI_HOST
  - QBITTORRENTAPI_USERNAME
  - QBITTORRENTAPI_PASSWORD

### qBittorrent Session Management

- Any time a connection is established with qBittorrent, it instantiates a session to manage authentication for all subsequent API requests.
- This client will transparently manage sessions by ensuring the client is always logged in in-line with any API request including requesting a new session upon expiration of an existing session.
- However, each new *Client* instantiation will create a new session in qBittorrent.
- Therefore, if many *Client* instances will be created be sure to call *auth_log_out* for each instance or use a context manager.
- Otherwise, qBittorrent may experience abnormally high memory usage.

```
with qbittorrentapi.Client(**conn_info) as qbt_client:
    if qbt_client.torrents_add(urls="...") != "Ok.":
        raise Exception("Failed to add torrent.")
```

### Untrusted WebUI Certificate

- qBittorrent allows you to configure HTTPS with an untrusted certificate; this commonly includes self-signed certificates.
- When using such a certificate, instantiate Client with `VERIFY_WEBUI_CERTIFICATE=False` or set environment variable `QBITTORRENTAPI_DO_NOT_VERIFY_WEBUI_CERTIFICATE` to a non-null value.
- Failure to do this for will cause connections to qBittorrent to fail.
- As a word of caution, doing this actually does turn off certificate verification. Therefore, for instance, potential man-in-the-middle attacks will not be detected and reported (since the error is suppressed). However, the connection will remain encrypted.

```
qbt_client = Client(..., VERIFY_WEBUI_CERTIFICATE=False}
```

### Requests Configuration

- The Requests package is used to issue HTTP requests to qBittorrent to facilitate this API.
- Much of *Requests* configuration for making HTTP requests can be controlled with parameters passed along with the request payload.
- For instance, HTTP Basic Authorization credentials can be provided via `auth`, timeouts via `timeout`, or Cookies via `cookies`. See Requests documentation for full details.
- These parameters are exposed here in two ways; the examples below tell `Requests` to use a connect timeout of 3.1 seconds and a read timeout of 30 seconds.
- When you instantiate *Client*, you can specify the parameters to use in all HTTP requests to qBittorrent:

```
qbt_client = Client(..., REQUESTS_ARGS={'timeout': (3.1, 30)}
```

- Alternatively, parameters can be specified for individual requests:

```
qbt_client.torrents_info(..., requests_args={'timeout': (3.1, 30)})
```

### Additional HTTP Headers

- For consistency, HTTP Headers can be specified using the method above; for backwards compatibility, the methods below are supported as well.
- Either way, these additional headers will be incorporated (using clobbering) into the rest of the headers to be sent.
- To send a custom HTTP header in all requests made from an instantiated client, declare them during instantiation:

```
qbt_client = Client(..., EXTRA_HEADERS={'X-My-Fav-Header': 'header value')
```

- Alternatively, you can send custom headers in individual requests:

```
qbt_client.torrents.add(..., headers={'X-My-Fav-Header': 'header value')
```

### Unimplemented API Endpoints

- Since the qBittorrent Web API has evolved over time, some endpoints may not be available from the qBittorrent host.
- By default, if a request is made to endpoint that doesn't exist for the version of the qBittorrent host (e.g., the Search endpoints were introduced in Web API v2.1.1), there's a debug logger output and None is returned.
- To raise `NotImplementedError` instead, instantiate Client with:

```
qbt_client = Client(..., RAISE_NOTIMPLEMENTEDERROR_FOR_UNIMPLEMENTED_API_ENDPOINTS=True)
```

### qBittorrent Version Checking

- It is also possible to either raise an Exception for qBittorrent hosts that are not "fully" supported or manually check for support.
- The most likely situation for this to occur is if the qBittorrent team publishes a new release but its changes have not been incorporated in to this client yet.
- Instantiate Client like below to raise `UnsupportedQbittorrentVersion` exception for versions not fully supported:

```
qbt_client = Client(..., RAISE_ERROR_FOR_UNSUPPORTED_QBITTORRENT_VERSIONS=True)
```

- Additionally, the *qbittorrentapi.Version* class can be used for manual introspection of the versions.

```
Version.is_app_version_supported(qbt_client.app.version)
```

#### Disable Logging Debug Output

- Instantiate Client with `DISABLE_LOGGING_DEBUG_OUTPUT=True` or manually disable logging for the relevant packages:

```
logging.getLogger('qbittorrentapi').setLevel(logging.INFO)
logging.getLogger('requests').setLevel(logging.INFO)
logging.getLogger('urllib3').setLevel(logging.INFO)
```

### 1.4.3 Performance

By default, complex objects are returned from some endpoints. These objects allow for accessing the response's items as attributes and include methods for contextually relevant actions (such as `start()` and `stop()` for a torrent, for example).

This comes at the cost of performance, though. Generally, this cost isn't large; however, some endpoints, such as `torrents_files()`, may need to convert a large payload and the cost can be significant.

This client can be configured to always return only the simple JSON if desired. Simply set `SIMPLE_RESPONSES=True` when instantiating the client.

```
qbt_client = qbittorrentapi.Client(
    host='localhost:8080',
    username='admin',
    password='adminadmin',
    SIMPLE_RESPONSES=True,
)
```

Alternatively, `SIMPLE_RESPONSES` can be set to `True` to return the simple JSON only for an individual method call.

```
qbt_client.torrents.files(torrent_hash='...', SIMPLE_RESPONSES=True)
```

### 1.4.4 Exceptions

**exception APIError**

Bases: Exception

Base error for all exceptions from this Client.

**exception UnsupportedQbittorrentVersion**

Bases: *APIError*

Connected qBittorrent is not fully supported by this Client.

**exception FileError**

Bases: OSError, *APIError*

Base class for all exceptions for file handling.

**exception TorrentFileError**

Bases: *FileError*

Base class for all exceptions for torrent files.

**exception TorrentFileNotFoundError**

Bases: *TorrentFileError*

Specified torrent file does not appear to exist.

**exception TorrentFilePermissionError**

Bases: *TorrentFileError*

Permission was denied to read the specified torrent file.

**exception APIConnectionError**(**args*, ***kwargs*)

Bases: RequestException, *APIError*

Base class for all communications errors including HTTP errors.

**exception LoginFailed**(**args*, ***kwargs*)

Bases: *APIConnectionError*

This can technically be raised with any request since log in may be attempted for any request and could fail.

**exception HTTPError**(**args*, ***kwargs*)

Bases: HTTPError, *APIConnectionError*

Base error for all HTTP errors.

All errors following a successful connection to qBittorrent are returned as HTTP statuses.

**http_status_code:** int

**exception HTTP4XXError**(**args*, ***kwargs*)

Bases: *HTTPError*

Base error for all HTTP 4XX statuses.

**exception HTTP5XXError**(**args*, ***kwargs*)

Bases: *HTTPError*

Base error for all HTTP 5XX statuses.

**exception HTTP400Error**(**args*, ***kwargs*)

Bases: *HTTP4XXError*

HTTP 400 Status.

**http_status_code:** int = 400

**exception HTTP401Error**(**args*, ***kwargs*)

Bases: *HTTP4XXError*

HTTP 401 Status.

**http_status_code:** int = 401

**exception HTTP403Error**(**args*, ***kwargs*)

Bases: *HTTP4XXError*

HTTP 403 Status.

**http_status_code:** int = 403

**exception HTTP404Error**(**args*, ***kwargs*)

Bases: *HTTP4XXError*

HTTP 404 Status.

**http_status_code:** int = 404

**exception HTTP405Error**(**args*, ***kwargs*)

Bases: *HTTP4XXError*

HTTP 405 Status.

**http_status_code:** int = 405

**exception HTTP409Error**(**args*, ***kwargs*)

Bases: *HTTP4XXError*

HTTP 409 Status.

**http_status_code:** int = 409

**exception HTTP415Error**(**args*, ***kwargs*)

Bases: *HTTP4XXError*

HTTP 415 Status.

**http_status_code:** int = 415

**exception HTTP500Error**(**args*, ***kwargs*)

Bases: *HTTP5XXError*

HTTP 500 Status.

**http_status_code:** int = 500

**exception MissingRequiredParameters400Error**(**args*, ***kwargs*)

Bases: *HTTP400Error*

Endpoint call is missing one or more required parameters.

**exception InvalidRequest400Error**(**args*, ***kwargs*)

Bases: *HTTP400Error*

One or more endpoint arguments are malformed.

**exception Unauthorized401Error**(**args*, ***kwargs*)

Bases: *HTTP401Error*

Primarily reserved for XSS and host header issues.

**exception Forbidden403Error**(**args*, ***kwargs*)

Bases: *HTTP403Error*

Not logged in, IP has been banned, or calling an API method that isn't public.

**exception NotFound404Error**(**args*, ***kwargs*)

Bases: *HTTP404Error*

This should mean qBittorrent couldn't find a torrent for the torrent hash.

**exception MethodNotAllowed405Error**(**args*, ***kwargs*)

Bases: *HTTP405Error*

HTTP method is not supported for the API endpoint.

**exception Conflict409Error**(**args*, ***kwargs*)

Bases: *HTTP409Error*

Returned if arguments don't make sense specific to the endpoint.

**exception UnsupportedMediaType415Error**(**args*, ***kwargs*)

Bases: *HTTP415Error*

`torrents/add` endpoint will return this for invalid URL(s) or files.

**exception InternalServerError500Error**(**args*, ***kwargs*)

Bases: *HTTP500Error*

Returned if qBittorrent errors internally while processing the request.

### 1.4.5 API Reference

#### Application

**class AppAPIMixIn**(*host=None*, *port=None*, *username=None*, *password=None*, *EXTRA_HEADERS=None*, *REQUESTS_ARGS=None*, *VERIFY_WEBUI_CERTIFICATE=True*, *FORCE_SCHEME_FROM_HOST=False*, *RAISE_NOTIMPLEMENTEDERROR_FOR_UNIMPLEMENTED_API_ENDPOINTS=False*, *RAISE_ERROR_FOR_UNSUPPORTED_QBITTORRENT_VERSIONS=False*, *VERBOSE_RESPONSE_LOGGING=False*, *SIMPLE_RESPONSES=False*, *DISABLE_LOGGING_DEBUG_OUTPUT=False*) → None

Bases: *AuthAPIMixIn*

Implementation of all `Application` API methods.

**Usage**

```
>>> from qbittorrentapi import Client
>>> client = Client(host="localhost:8080", username="admin", password=
↪"adminadmin")
>>> client.app_version()
>>> client.app_preferences()
```

**app_build_info**(***kwargs*) → *BuildInfoDictionary*

qBittorrent build info.

This method was introduced with qBittorrent v4.2.0 (Web API v2.3).

**Return type**
*BuildInfoDictionary*

**app_default_save_path**(***kwargs*) → str

The default path where torrents are saved.

**Return type**
str

**app_network_interface_address_list**(*interface_name=''*, ***kwargs*) → *NetworkInterfaceAddressList*

The addresses for a network interface; omit name for all addresses.

This method was introduced with qBittorrent v4.2.0 (Web API v2.3).

**Parameters**
**interface_name** (str) – Name of interface to retrieve addresses for

**Return type**
*NetworkInterfaceAddressList*

**app_network_interface_list**(***kwargs*) → *NetworkInterfaceList*

The list of network interfaces on the host.

This method was introduced with qBittorrent v4.2.0 (Web API v2.3).

**Return type**
*NetworkInterfaceList*

**app_preferences**(***kwargs*) → *ApplicationPreferencesDictionary*

Retrieve qBittorrent application preferences.

**Return type**
*ApplicationPreferencesDictionary*

**app_set_preferences**(*prefs=None*, ***kwargs*) → None

Set one or more preferences in qBittorrent application.

**Parameters**
**prefs** (Optional[Mapping[str, Any]]) – dictionary of preferences to set

**Return type**
None

**app_shutdown**(***kwargs*) → None

Shutdown qBittorrent.

**Return type**
None

**app_version**(***kwargs*) → str

qBittorrent application version.

**Return type**
str

**app_web_api_version**(***kwargs*) → str

qBittorrent Web API version.

**Return type**
str

**class Application**(**args*, *client*, ***kwargs*)

Allows interaction with `Application` API endpoints.

**Usage**

```
>>> from qbittorrentapi import Client
>>> client = Client(host="localhost:8080", username="admin", password=
↪"adminadmin")
>>> # these are all the same attributes that are available as named in␣
```

(continues on next page)

(continued from previous page)

```
↪the
>>> #  endpoints or the more pythonic names in Client (with or without
↪'app_' prepended)
>>> webapiVersion = client.application.webapiVersion
>>> web_api_version = client.application.web_api_version
>>> app_web_api_version = client.application.web_api_version
>>> # access and set preferences as attributes
>>> is_dht_enabled = client.application.preferences.dht
>>> # supports sending a just subset of preferences to update
>>> client.application.preferences = dict(dht=(not is_dht_enabled))
>>> prefs = client.application.preferences
>>> prefs["web_ui_clickjacking_protection_enabled"] = True
>>> client.app.preferences = prefs
>>>
>>> client.application.shutdown()
```

**property build_info:** *BuildInfoDictionary*

qBittorrent build info.

This method was introduced with qBittorrent v4.2.0 (Web API v2.3).

**property default_save_path:** str

The default path where torrents are saved.

**network_interface_address_list**(*interface_name=''*, ***kwargs*) → *NetworkInterfaceAddressList*

The addresses for a network interface; omit name for all addresses.

This method was introduced with qBittorrent v4.2.0 (Web API v2.3).

**Parameters**
**interface_name** (str) – Name of interface to retrieve addresses for

**Return type**
*NetworkInterfaceAddressList*

**property network_interface_list:** *NetworkInterfaceList*

The list of network interfaces on the host.

This method was introduced with qBittorrent v4.2.0 (Web API v2.3).

**property preferences:** *ApplicationPreferencesDictionary*

Retrieve qBittorrent application preferences.

**set_preferences**(*prefs=None*, ***kwargs*) → None

Set one or more preferences in qBittorrent application.

**Parameters**
**prefs** (Optional[Mapping[str, Any]]) – dictionary of preferences to set

**Return type**
None

**shutdown**(***kwargs*) → None

Shutdown qBittorrent.

**Return type**
None

**property version: str**

qBittorrent application version.

**property web_api_version: str**

qBittorrent Web API version.

**class ApplicationPreferencesDictionary**(*data=None*, ***kwargs*)

Bases: *Dictionary*[Union[None, int, str, bool, Sequence[JsonValueT], Mapping[str, JsonValueT]]]

Response for *app_preferences()*

Definition: https://github.com/qbittorrent/qBittorrent/wiki/WebUI-API-(qBittorrent-4.1)#user-content-get-application-preferences

**class BuildInfoDictionary**(*data=None*, ***kwargs*)

Bases: *Dictionary*[Union[str, int]]

Response for *app_build_info()*

Definition: https://github.com/qbittorrent/qBittorrent/wiki/WebUI-API-(qBittorrent-4.1)#user-content-get-build-info

**class NetworkInterfaceList**(*list_entries*, *client=None*)

Bases: *List*[*NetworkInterface*]

Response for *app_network_interface_list()*

**class NetworkInterface**(*data=None*, ***kwargs*)

Bases: *ListEntry*

Item in *NetworkInterfaceList*

**class NetworkInterfaceAddressList**(*list_entries*, *client=None*)

Bases: *List*[str]

Response for *app_network_interface_address_list()*

### AttrDict (internal)

Copyright (c) 2013 Brendan Curran-Johnson

**class AttrDict**(**args*, ***kwargs*) → None

Bases: Dict[str, V], *MutableAttr*[V]

A dict that implements MutableAttr.

**class MutableAttr**

Bases: *Attr*[str, V], MutableMapping[str, V], ABC

A `mixin` mapping class that allows for attribute-style access of values.

**class Attr**

Bases: Mapping[K, V], ABC

A `mixin` class for a mapping that allows for attribute-style access of values.

**A key may be used as an attribute if:**

- It is a string
- It matches `^[A-Za-z][A-Za-z0-9_]*$` (i.e., a public attribute)

- The key doesn't overlap with any class attributes (for `Attr`, those would be `get`, `items`, `keys`, `values`, `mro`, and `register`).

If a value which is accessed as an attribute is a Sequence-type (and is not a string/bytes), it will be converted to a _sequence_type with any mappings within it converted to Attrs.

**NOTE:**
This means that if _sequence_type is not None, then a sequence accessed as an attribute will be a different object than if accessed as an attribute than if it is accessed as an item.

### Authentication

**class AuthAPIMixIn**(*host=None*, *port=None*, *username=None*, *password=None*, *EXTRA_HEADERS=None*, *REQUESTS_ARGS=None*, *VERIFY_WEBUI_CERTIFICATE=True*, *FORCE_SCHEME_FROM_HOST=False*, *RAISE_NOTIMPLEMENTEDERROR_FOR_UNIMPLEMENTED_API_ENDPOINTS=False*, *RAISE_ERROR_FOR_UNSUPPORTED_QBITTORRENT_VERSIONS=False*, *VERBOSE_RESPONSE_LOGGING=False*, *SIMPLE_RESPONSES=False*, *DISABLE_LOGGING_DEBUG_OUTPUT=False*) → None

Bases: *Request*

Implementation of all `Authorization` API methods.

**Usage**

```
>>> from qbittorrentapi import Client
>>> client = Client(host="localhost:8080", username="admin", password=
↪"adminadmin")
>>> _ = client.is_logged_in
>>> client.auth_log_in(username="admin", password="adminadmin")
>>> client.auth_log_out()
```

**auth_log_in**(*username=None*, *password=None*, ***kwargs*) → None

Log in to qBittorrent host.

**Raises**

- ***LoginFailed*** – if credentials failed to log in
- ***Forbidden403Error*** – if user is banned…or not logged in

**Parameters**

- **username** (Optional[str]) – username for qBittorrent client
- **password** (Optional[str]) – password for qBittorrent client

**Return type**
None

**auth_log_out**(***kwargs*) → None

End session with qBittorrent.

**Return type**
None

**property is_logged_in: bool**

Returns True if low-overhead API call succeeds; False otherwise.

There isn't a reliable way to know if an existing session is still valid without attempting to use it. qBittorrent invalidates cookies when they expire.

**Returns**
`True`/`False` if current authorization cookie is accepted by qBittorrent

**class Authorization**(**args*, *client*, ***kwargs*)

Allows interaction with the `Authorization` API endpoints.

**Usage**

```
>>> from qbittorrentapi import Client
>>> client = Client(host="localhost:8080", username="admin", password=
↪"adminadmin")
>>> is_logged_in = client.auth.is_logged_in
>>> client.auth.log_in(username="admin", password="adminadmin")
>>> client.auth.log_out()
```

**property is_logged_in**

Returns True if low-overhead API call succeeds; False otherwise.

There isn't a reliable way to know if an existing session is still valid without attempting to use it. qBittorrent invalidates cookies when they expire.

**Returns**
`True`/`False` if current authorization cookie is accepted by qBittorrent

**log_in**(*username=None*, *password=None*, ***kwargs*) → None

Log in to qBittorrent host.

**Raises**

- ***LoginFailed*** – if credentials failed to log in
- ***Forbidden403Error*** – if user is banned…or not logged in

**Parameters**

- **username** (Optional[str]) – username for qBittorrent client
- **password** (Optional[str]) – password for qBittorrent client

**Return type**
None

**log_out**(***kwargs*) → None

End session with qBittorrent.

**Return type**
None

### Client

class Client(*host=''*, *port=None*, *username=None*, *password=None*, *EXTRA_HEADERS=None*, *REQUESTS_ARGS=None*, *VERIFY_WEBUI_CERTIFICATE=True*, *FORCE_SCHEME_FROM_HOST=False*, *RAISE_NOTIMPLEMENTEDERROR_FOR_UNIMPLEMENTED_API_ENDPOINTS=False*, *RAISE_ERROR_FOR_UNSUPPORTED_QBITTORRENT_VERSIONS=False*, *VERBOSE_RESPONSE_LOGGING=False*, *SIMPLE_RESPONSES=False*, *DISABLE_LOGGING_DEBUG_OUTPUT=False*)

Bases: *LogAPIMixIn*, *SyncAPIMixIn*, *TransferAPIMixIn*, *TorrentsAPIMixIn*, *RSSAPIMixIn*, *SearchAPIMixIn*

Initialize API for qBittorrent client.

Host must be specified. Username and password can be specified at login. A call to *auth_log_in()* is not explicitly required if username and password are provided during Client construction.

**Usage**

```
>>> from qbittorrentapi import Client
>>> client = Client(host='localhost:8080', username='admin', password=
↪'adminadmin')
>>> torrents = client.torrents_info()
```

**Parameters**

- **host** (str) – hostname for qBittorrent Web API, `[http[s]://]localhost[:8080][/path]`
- **port** (UnionType[str, int, None]) – port number for qBittorrent Web API (ignored if host contains a port)
- **username** (Optional[str]) – username for qBittorrent Web API
- **password** (Optional[str]) – password for qBittorrent Web API
- **SIMPLE_RESPONSES** (bool) – By default, complex objects are returned from some endpoints. These objects will allow for accessing responses' items as attributes and include methods for contextually relevant actions. This comes at the cost of performance. Generally, this cost isn't large; however, some endpoints, such as `torrents_files()` method, may need to convert a large payload. Set this to True to return the simple JSON back. Alternatively, set this to True only for an individual method call. For instance, when requesting the files for a torrent: `client.torrents_files(torrent_hash='...', SIMPLE_RESPONSES=True)`
- **VERIFY_WEBUI_CERTIFICATE** (bool) – Set to False to skip verify certificate for HTTPS connections; for instance, if the connection is using a self-signed certificate. Not setting this to False for self-signed certs will cause a *APIConnectionError* exception to be raised.
- **EXTRA_HEADERS** (Optional[Mapping[str, str]]) – Dictionary of HTTP Headers to include in all requests made to qBittorrent.
- **REQUESTS_ARGS** (Optional[Mapping[str, Any]]) – Dictionary of configuration for Requests package: https://requests.readthedocs.io/en/latest/api/#requests.request
- **FORCE_SCHEME_FROM_HOST** (bool) – If a scheme (i.e. `http` or `https`) is specified in host, it will be used regardless of whether qBittorrent is configured for HTTP or HTTPS communication. Normally, this client will attempt to determine which scheme qBittorrent is actually listening on… but this can cause problems in rare cases. Defaults `False`.

- **RAISE_NOTIMPLEMENTEDERROR_FOR_UNIMPLEMENTED_API_ENDPOINTS** (bool) – Some Endpoints may not be implemented in older versions of qBittorrent. Setting this to True will raise a NotImplementedError instead of just returning``None``.
- **RAISE_ERROR_FOR_UNSUPPORTED_QBITTORRENT_VERSIONS** (bool) – raise the UnsupportedQbittorrentVersion exception if the connected version of qBittorrent is not fully supported by this client. Defaults False.
- **DISABLE_LOGGING_DEBUG_OUTPUT** (bool) – Turn off debug output from logging for this package as well as Requests & urllib3.

### Definitions

**APIKwargsT**

Type Any for kwargs parameters for API methods.

**class APINames**(*value*, *names=None*, ***, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

Bases: str, Enum

API namespaces for API endpoints.

e.g torrents in http://localhost:8080/api/v2/torrents/addTrackers

**Application = 'app'**

**Authorization = 'auth'**

**EMPTY = ''**

**Log = 'log'**

**RSS = 'rss'**

**Search = 'search'**

**Sync = 'sync'**

**Torrents = 'torrents'**

**Transfer = 'transfer'**

**class ClientCache**(**args*, *client*, ***kwargs*)

Bases: Generic[*ClientT*]

Caches the client.

Subclass this for any object that needs access to the Client.

**class ClientT**

Type for this API Client.

alias of TypeVar('ClientT', bound=Request)

**class Dictionary**(*data=None*, ***kwargs*)

Bases: *AttrDict*[V]

Base definition of dictionary-like objects returned from qBittorrent.

**FilesToSendT**

Type for Files input to API method.

alias of Mapping[str, Union[bytes, Tuple[str, bytes]]]

**JsonValueT**

Type to define JSON.

alias of Union[None, int, str, bool, Sequence[*JsonValueT*], Mapping[str, *JsonValueT*]]

**class List**(*list_entries=None*, *entry_class=None*, ***kwargs*)

Bases: UserList[*ListEntryT*]

Base definition for list-like objects returned from qBittorrent.

**class ListEntry**(*data=None*, ***kwargs*)

Bases: *Dictionary*[Union[None, int, str, bool, Sequence[*JsonValueT*], Mapping[str, *JsonValueT*]]]

Base definition for objects within a list returned from qBittorrent.

**class ListEntryT**

Type for entry in List from API.

alias of TypeVar('ListEntryT', bound=*ListEntry*)

**ListInputT**

Type for List input to API method.

alias of Iterable[Mapping[str, Union[None, int, str, bool, Sequence[*JsonValueT*], Mapping[str, *JsonValueT*]]]]

**class TorrentState**(*value*, *names=None*, ***, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

Bases: str, Enum

Torrent States as defined by qBittorrent.

**Definitions:**

- wiki: https://github.com/qbittorrent/qBittorrent/wiki/WebUI-API-(qBittorrent-4.1)#user-content-get-torrent-list
- code: https://github.com/qbittorrent/qBittorrent/blob/5dcc14153f046209f1067299494a82e5294d883a/src/base/bittorrent/torrent.h#L73

**Usage**

```
>>> from qbittorrentapi import Client, TorrentState
>>> client = Client()
>>> # print torrent hashes for torrents that are downloading
>>> for torrent in client.torrents_info():
>>>     # check if torrent is downloading
>>>     if torrent.state_enum.is_downloading:
>>>         print(f'{torrent.hash} is downloading...')
>>>     # the appropriate enum member can be directly derived
>>>     state_enum = TorrentState(torrent.state)
>>>     print(f'{torrent.hash}: {state_enum.value}')
```

**ALLOCATING = 'allocating'**

CHECKING_DOWNLOAD = 'checkingDL'

CHECKING_RESUME_DATA = 'checkingResumeData'

CHECKING_UPLOAD = 'checkingUP'

DOWNLOADING = 'downloading'

ERROR = 'error'

FORCED_DOWNLOAD = 'forcedDL'

FORCED_METADATA_DOWNLOAD = 'forcedMetaDL'

FORCED_UPLOAD = 'forcedUP'

METADATA_DOWNLOAD = 'metaDL'

MISSING_FILES = 'missingFiles'

MOVING = 'moving'

PAUSED_DOWNLOAD = 'pausedDL'

PAUSED_UPLOAD = 'pausedUP'

QUEUED_DOWNLOAD = 'queuedDL'

QUEUED_UPLOAD = 'queuedUP'

STALLED_DOWNLOAD = 'stalledDL'

STALLED_UPLOAD = 'stalledUP'

UNKNOWN = 'unknown'

UPLOADING = 'uploading'

property is_checking: bool
: Returns True if the State is categorized as Checking.

property is_complete: bool
: Returns True if the State is categorized as Complete.

property is_downloading: bool
: Returns True if the State is categorized as Downloading.

property is_errored: bool
: Returns True if the State is categorized as Errored.

property is_paused: bool
: Returns True if the State is categorized as Paused.

property is_uploading: bool
: Returns True if the State is categorized as Uploading.

class TrackerStatus(*value*, *names=None*, ***, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

Bases: int, Enum

Tracker Statuses as defined by qBittorrent.

**Definitions:**

- wiki: https://github.com/qbittorrent/qBittorrent/wiki/WebUI-API-(qBittorrent-4.1)#user-content-get-torrent-trackers
- code: https://github.com/qbittorrent/qBittorrent/blob/5dcc14153f046209f1067299494a82e5294d883a/src/base/bittorrent/trackerentry.h#L42

**Usage**

```
>>> from qbittorrentapi import Client, TrackerStatus
>>> client = Client()
>>> # print torrent hashes for torrents that are downloading
>>> for torrent in client.torrents_info():
>>>     for tracker in torrent.trackers:
>>>         # display status for each tracker
>>>         print(f"{torrent.hash[-6:]}: {TrackerStatus(tracker.status).
↪display:>13} :{tracker.url}")
```

**DISABLED = 0**

**NOT_CONTACTED = 1**

**NOT_WORKING = 4**

**UPDATING = 3**

**WORKING = 2**

**property display: str**

Returns a descriptive display value for status.

## Log

class LogAPIMixIn(*host=None*, *port=None*, *username=None*, *password=None*, *EXTRA_HEADERS=None*, *REQUESTS_ARGS=None*, *VERIFY_WEBUI_CERTIFICATE=True*, *FORCE_SCHEME_FROM_HOST=False*, *RAISE_NOTIMPLEMENTEDERROR_FOR_UNIMPLEMENTED_API_ENDPOINTS=False*, *RAISE_ERROR_FOR_UNSUPPORTED_QBITTORRENT_VERSIONS=False*, *VERBOSE_RESPONSE_LOGGING=False*, *SIMPLE_RESPONSES=False*, *DISABLE_LOGGING_DEBUG_OUTPUT=False*) → None

Bases: *AppAPIMixIn*

Implementation of all Log API methods.

**Usage**

```
>>> from qbittorrentapi import Client
>>> client = Client(host="localhost:8080", username="admin", password=
↪"adminadmin")
```

(continues on next page)

(continued from previous page)

```
>>> client.log_main(info=False)
>>> client.log_peers()
```

**log_main**(*normal=None*, *info=None*, *warning=None*, *critical=None*, *last_known_id=None*, ***kwargs*) → *LogMainList*

Retrieve the qBittorrent log entries. Iterate over returned object.

**Parameters**

- **normal** (Optional[bool]) – False to exclude `normal` entries
- **info** (Optional[bool]) – False to exclude `info` entries
- **warning** (Optional[bool]) – False to exclude `warning` entries
- **critical** (Optional[bool]) – False to exclude `critical` entries
- **last_known_id** (UnionType[str, int, None]) – only entries with an ID greater than this value will be returned

**Return type**
*LogMainList*

**log_peers**(*last_known_id=None*, ***kwargs*) → *LogPeersList*

Retrieve qBittorrent peer log.

**Parameters**
**last_known_id** (UnionType[str, int, None]) – only entries with an ID greater than this value will be returned

**Return type**
*LogPeersList*

**class Log**(*client*)

Allows interaction with Log API endpoints.

**Usage**

```
>>> from qbittorrentapi import Client
>>> client = Client(host="localhost:8080", username="admin", password=
↪"adminadmin")
>>> # this is all the same attributes that are available as named in the
>>> #  endpoints or the more pythonic names in Client (with or without
↪'log_' prepended)
>>> log_list = client.log.main()
>>> peers_list = client.log.peers(last_known_id="...")
>>> # can also filter log down with additional attributes
>>> log_info = client.log.main.info(last_known_id=1)
>>> log_warning = client.log.main.warning(last_known_id=1)
```

**peers**(*last_known_id=None*, ***kwargs*) → *LogPeersList*

Retrieve qBittorrent peer log.

**Parameters**
**last_known_id** (UnionType[str, int, None]) – only entries with an ID greater than this value will be returned

**Return type**
*LogPeersList*

**class LogPeersList**(*list_entries*, *client=None*)

Bases: *List*[*LogPeer*]

Response for *log_peers()*

**class LogPeer**(*data=None*, ***kwargs*)

Bases: *ListEntry*

Item in *LogPeersList*

**class LogMainList**(*list_entries*, *client=None*)

Bases: *List*[*LogEntry*]

Response to *log_main()*

**class LogEntry**(*data=None*, ***kwargs*)

Bases: *ListEntry*

Item in *LogMainList*

### Request (internal)

**class QbittorrentSession**

Bases: Session

Wrapper to augment Requests Session.

Requests doesn't allow Session to default certain configuration globally. This gets around that by setting defaults for each request.

**request**(*method*, *url*, *params=None*, *data=None*, *headers=None*, *cookies=None*, *files=None*, *auth=None*, *timeout=None*, *allow_redirects=True*, *proxies=None*, *hooks=None*, *stream=None*, *verify=None*, *cert=None*, *json=None*)

Constructs a *Request*, prepares it and sends it. Returns Response object.

**Parameters**

- **method** – method for the new *Request* object.
- **url** – URL for the new *Request* object.
- **params** – (optional) Dictionary or bytes to be sent in the query string for the *Request*.
- **data** – (optional) Dictionary, list of tuples, bytes, or file-like object to send in the body of the *Request*.
- **json** – (optional) json to send in the body of the *Request*.
- **headers** – (optional) Dictionary of HTTP Headers to send with the *Request*.
- **cookies** – (optional) Dict or CookieJar object to send with the *Request*.
- **files** – (optional) Dictionary of `'filename': file-like-objects` for multipart encoding upload.
- **auth** – (optional) Auth tuple or callable to enable Basic/Digest/Custom HTTP Auth.
- **timeout** (*float or tuple*) – (optional) How long to wait for the server to send data before giving up, as a float, or a (connect timeout, read timeout) tuple.
- **allow_redirects** (*bool*) – (optional) Set to True by default.

- **proxies** – (optional) Dictionary mapping protocol or protocol and hostname to the URL of the proxy.
- **stream** – (optional) whether to immediately download the response content. Defaults to `False`.
- **verify** – (optional) Either a boolean, in which case it controls whether we verify the server's TLS certificate, or a string, in which case it must be a path to a CA bundle to use. Defaults to `True`. When set to `False`, requests will accept any TLS certificate presented by the server, and will ignore hostname mismatches and/or expired certificates, which will make your application vulnerable to man-in-the-middle (MitM) attacks. Setting verify to `False` may be useful during local development or testing.
- **cert** – (optional) if String, path to ssl client cert file (.pem). If Tuple, ('cert', 'key') pair.

**Return type**
*Response*

`class QbittorrentURL(client)`

Bases: `object`

Management for the qBittorrent Web API URL.

`build(api_namespace, api_method, headers=None, requests_kwargs=None, base_path='api/v2')` → str

Create a fully qualified URL for the API endpoint.

**Parameters**

- **api_namespace** (*APINames* | `str`) – the namespace for the API endpoint (e.g. `torrents`)
- **api_method** (`str`) – the specific method for the API endpoint (e.g. `info`)
- **base_path** (`str`) – base path for URL (e.g. `api/v2`)
- **headers** (`Optional[Mapping[str, str]]`) – HTTP headers for request
- **requests_kwargs** (`Optional[Mapping[str, Any]]`) – kwargs for any calls to Requests

**Return type**
`str`

**Returns**
fully qualified URL string for endpoint

`build_base_url(headers, requests_kwargs)` → str

Determine the Base URL for the Web API endpoints.

A URL is only actually built here if it's the first time here or the context was re-initialized. Otherwise, the most recently built URL is used.

If the user doesn't provide a scheme for the URL, it will try HTTP first and fall back to HTTPS if that doesn't work. While this is probably backwards, qBittorrent or an intervening proxy can simply redirect to HTTPS and that'll be respected.

Additionally, if users want to augment the path to the API endpoints, any path provided here will be preserved in the returned Base URL and prefixed to all subsequent API calls.

**Parameters**

- **headers** (`Mapping[str, str]`) – HTTP headers for request
- **requests_kwargs** (`Mapping[str, Any]`) – arguments from user for HTTP `HEAD` request

**Return type**
`str`

**Returns**
base URL as a `string` for Web API endpoint

**detect_scheme**(*base_url*, *default_scheme*, *alt_scheme*, *headers*, *requests_kwargs*) → str

Determine if the URL endpoint is using HTTP or HTTPS.

**Parameters**

- **base_url** (ParseResult) – urllib `ParseResult` URL object
- **default_scheme** (str) – default scheme to use for URL
- **alt_scheme** (str) – alternative scheme to use for URL if default doesn't work
- **headers** (Mapping[str, str]) – HTTP headers for request
- **requests_kwargs** (Mapping[str, Any]) – kwargs for calls to Requests

**Return type**
str

**Returns**
scheme (i.e. `HTTP` or `HTTPS`)

**class Request**(*host=None*, *port=None*, *username=None*, *password=None*, *EXTRA_HEADERS=None*, *REQUESTS_ARGS=None*, *VERIFY_WEBUI_CERTIFICATE=True*, *FORCE_SCHEME_FROM_HOST=False*, *RAISE_NOTIMPLEMENTEDERROR_FOR_UNIMPLEMENTED_API_ENDPOINTS=False*, *RAISE_ERROR_FOR_UNSUPPORTED_QBITTORRENT_VERSIONS=False*, *VERBOSE_RESPONSE_LOGGING=False*, *SIMPLE_RESPONSES=False*, *DISABLE_LOGGING_DEBUG_OUTPUT=False*) → None

Bases: object

Facilitates HTTP requests to qBittorrent's Web API.

**_auth_request**(*http_method*, *api_namespace*, *api_method*, *_retry_backoff_factor=0.3*, *requests_args=None*, *requests_params=None*, *headers=None*, *params=None*, *data=None*, *files=None*, *response_class=<class 'requests.models.Response'>*, *version_introduced=''*, *version_removed=''*, ***kwargs*) → Any

Wraps API call with re-authorization if first attempt is not authorized.

**Return type**
Any

**_cast**(*response*, *response_class*, ***response_kwargs*) → Any

Returns the API response casted to the requested class.

**Parameters**

- **response** (Response) – requests `Response` from API
- **response_class** (type) – class to return response as; if none, response is returned
- **response_kwargs** (Any) – request-specific configuration for response

**Return type**
Any

**static _format_payload**(*http_method*, *params=None*, *data=None*, *files=None*, ***kwargs*) → tuple[dict[str, Any], dict[str, Any], Mapping[str, bytes | Tuple[str, bytes]]]

Determine `data`, `params`, and `files` for the Requests call.

**Parameters**

- **http_method** (str) – `get` or `post`
- **params** (Optional[Mapping[str, Any]]) – key value pairs to send with GET calls
- **data** (Optional[Mapping[str, Any]]) – key value pairs to send with POST calls
- **files** (Optional[Mapping[str, Union[bytes, Tuple[str, bytes]]]]) – dictionary of files to send with request

**Return type**
tuple[dict[str, Any], dict[str, Any], Mapping[str, Union[bytes, Tuple[str, bytes]]]]

**_get**(*_name*, *_method*, *requests_args=None*, *requests_params=None*, *headers=None*, *params=None*, *data=None*, *files=None*, *response_class=<class 'requests.models.Response'>*, *version_introduced=''*, *version_removed=''*, ***kwargs*) → Any

Send `GET` request.

**Parameters**

- **api_namespace** – the namespace for the API endpoint (e.g. *APINames* or `torrents`)
- **api_method** – the name for the API endpoint (e.g. `add`)
- **kwargs** (Any) – see *_request()*

**Return type**
Any

**Returns**
Requests `Response`

**_get_cast**(*_name*, *_method*, *response_class*, *requests_args=None*, *requests_params=None*, *headers=None*, *params=None*, *data=None*, *files=None*, *version_introduced=''*, *version_removed=''*, ***kwargs*) → ResponseT

Send `GET` request with casted response.

**Parameters**

- **api_namespace** – the namespace for the API endpoint (e.g. *APINames* or `torrents`)
- **api_method** – the name for the API endpoint (e.g. `add`)
- **kwargs** (Any) – see *_request()*

**Return type**
TypeVar(ResponseT)

**static _handle_error_responses**(*data*, *params*, *response*) → None

Raise proper exception if qBittorrent returns Error HTTP Status.

**Return type**
None

**_initialize_context**() → None

Initialize and reset communications context with qBittorrent.

This is necessary on startup or when the authorization cookie needs to be replaced…perhaps because it expired, qBittorrent was restarted, significant settings changes, etc.

**Return type**
None

**_initialize_settings**(*EXTRA_HEADERS=None*, *REQUESTS_ARGS=None*, *VERIFY_WEBUI_CERTIFICATE=True*, *FORCE_SCHEME_FROM_HOST=False*, *RAISE_NOTIMPLEMENTEDERROR_FOR_UNIMPLEMENTED_API_ENDPOINTS=False*, *RAISE_ERROR_FOR_UNSUPPORTED_QBITTORRENT_VERSIONS=False*, *VERBOSE_RESPONSE_LOGGING=False*, *SIMPLE_RESPONSES=False*, *DISABLE_LOGGING_DEBUG_OUTPUT=False*) → None

Initialize lesser used configuration.

**Return type**
None

**_is_endpoint_supported_for_version**(*endpoint*, *version_introduced*, *version_removed*) → bool

Prevent using an API methods that doesn't exist in this version of qBittorrent.

**Parameters**

- **endpoint** (str) – name of the removed endpoint, e.g. torrents/ban_peers
- **version_introduced** (str) – the Web API version the endpoint was introduced
- **version_removed** (str) – the Web API version the endpoint was removed

**Return type**
bool

**classmethod _list2string**(*input_list*, *delimiter='|'*) → str | T

Convert entries in a list to a concatenated string.

**Parameters**

- **input_list** (TypeVar(T)) – list to convert
- **delimiter** (str) – delimiter for concatenation

**Return type**
Union[str, TypeVar(T)]

**_post**(*_name*, *_method*, *requests_args=None*, *requests_params=None*, *headers=None*, *params=None*, *data=None*, *files=None*, *response_class=<class 'requests.models.Response'>*, *version_introduced=''*, *version_removed=''*, ***kwargs*) → Any

Send POST request.

**Parameters**

- **api_namespace** – the namespace for the API endpoint (e.g. *APINames* or torrents)
- **api_method** – the name for the API endpoint (e.g. add)
- **kwargs** (Any) – see *_request()*

**Return type**
Any

**_post_cast**(*_name*, *_method*, *response_class*, *requests_args=None*, *requests_params=None*, *headers=None*, *params=None*, *data=None*, *files=None*, *version_introduced=''*, *version_removed=''*, ***kwargs*) → ResponseT

Send POST request with casted response.

**Parameters**

- **api_namespace** – the namespace for the API endpoint (e.g. *APINames* or torrents)
- **api_method** – the name for the API endpoint (e.g. add)

- **kwargs** (Any) – see *_request()*

**Return type**
TypeVar(ResponseT)

**_request**(*http_method*, *api_namespace*, *api_method*, *requests_args=None*, *requests_params=None*, *headers=None*, *params=None*, *data=None*, *files=None*, *response_class=<class 'requests.models.Response'>*, ***kwargs*) → Any

Meat and potatoes of sending requests to qBittorrent.

**Parameters**

- **http_method** (str) – `get` or `post`
- **api_namespace** (*APINames* | str) – the namespace for the API endpoint (e.g. *APINames* or `torrents`)
- **api_method** (str) – the name for the API endpoint (e.g. `add`)
- **requests_args** (Optional[Mapping[str, Any]]) – default location for Requests kwargs
- **requests_params** (Optional[Mapping[str, Any]]) – alternative location for Requests kwargs
- **headers** (Optional[Mapping[str, str]]) – HTTP headers to send with the request
- **params** (Optional[Mapping[str, Any]]) – key/value pairs to send with a `GET` request
- **data** (Optional[Mapping[str, Any]]) – key/value pairs to send with a `POST` request
- **files** (Optional[Mapping[str, Union[bytes, Tuple[str, bytes]]]]) – files to be sent with the request
- **response_class** (type) – class to use to cast the API response
- **kwargs** (Any) – arbitrary keyword arguments to send with the request

**Return type**
Any

**_request_manager**(*http_method*, *api_namespace*, *api_method*, *_retries=1*, *_retry_backoff_factor=0.3*, *requests_args=None*, *requests_params=None*, *headers=None*, *params=None*, *data=None*, *files=None*, *response_class=<class 'requests.models.Response'>*, *version_introduced=''*, *version_removed=''*, ***kwargs*) → Any

Wrapper to manage request retries and severe exceptions.

This should retry at least once to account for the Web API switching from HTTP to HTTPS. During the second attempt, the URL is rebuilt using HTTP or HTTPS as appropriate.

**Return type**
Any

**property _session:** *QbittorrentSession*

Create or return existing HTTP session.

**_trigger_session_initialization**() → None

Effectively resets the HTTP session by removing the reference to it.

During the next request, a new session will be created.

**Return type**
None

**_verbose_logging**(*url*, *data*, *params*, *requests_kwargs*, *response*) → None

Log verbose information about request; can be useful during development.

**Return type**
None

## RSS

**class RSSAPIMixIn**(*host=None*, *port=None*, *username=None*, *password=None*, *EXTRA_HEADERS=None*, *REQUESTS_ARGS=None*, *VERIFY_WEBUI_CERTIFICATE=True*, *FORCE_SCHEME_FROM_HOST=False*, *RAISE_NOTIMPLEMENTEDERROR_FOR_UNIMPLEMENTED_API_ENDPOINTS=False*, *RAISE_ERROR_FOR_UNSUPPORTED_QBITTORRENT_VERSIONS=False*, *VERBOSE_RESPONSE_LOGGING=False*, *SIMPLE_RESPONSES=False*, *DISABLE_LOGGING_DEBUG_OUTPUT=False*) → None

Bases: *AppAPIMixIn*

Implementation of all RSS API methods.

**Usage**

```
>>> from qbittorrentapi import Client
>>> client = Client(host="localhost:8080", username="admin", password=
↪"adminadmin")
>>> rss_rules = client.rss_rules()
>>> client.rss_set_rule(rule_name="...", rule_def={...})
```

**rss_add_feed**(*url=None*, *item_path=None*, ***kwargs*) → None

Add new RSS feed. Folders in path must already exist.

**Raises**
*Conflict409Error* –

**Parameters**

- **url** (Optional[str]) – URL of RSS feed (e.g. https://distrowatch.com/news/torrents.xml)
- **item_path** (Optional[str]) – Name and/or path for new feed (e.g. Folder\Subfolder\FeedName)

**Return type**
None

**rss_add_folder**(*folder_path=None*, ***kwargs*) → None

Add an RSS folder. Any intermediate folders in path must already exist.

**Raises**
*Conflict409Error* –

**Parameters**
**folder_path** (Optional[str]) – path to new folder (e.g. Linux\ISOs)

**Return type**
None

**rss_items**(*include_feed_data=None*, ***kwargs*) → *RSSitemsDictionary*

Retrieve RSS items and optionally feed data.

**Parameters**
**include_feed_data** (Optional[bool]) – True or false to include feed data

**Return type**
*RSSitemsDictionary*

**rss_mark_as_read**(*item_path=None*, *article_id=None*, ***kwargs*) → None

Mark RSS article as read. If article ID is not provider, the entire feed is marked as read.

This method was introduced with qBittorrent v4.2.5 (Web API v2.5.1).

**Raises**
*NotFound404Error* –

**Parameters**

- **item_path** (Optional[str]) – path to item to be refreshed (e.g. `Folder\Subfolder\ItemName`)
- **article_id** (UnionType[str, int, None]) – article ID from *rss_items()*

**Return type**
None

**rss_matching_articles**(*rule_name=None*, ***kwargs*) → *RSSitemsDictionary*

Fetch all articles matching a rule.

This method was introduced with qBittorrent v4.2.5 (Web API v2.5.1).

**Parameters**
**rule_name** (Optional[str]) – Name of rule to return matching articles

**Return type**
*RSSitemsDictionary*

**rss_move_item**(*orig_item_path=None*, *new_item_path=None*, ***kwargs*) → None

Move/rename an RSS item (folder, feed, etc.).

**Raises**
*Conflict409Error* –

**Parameters**

- **orig_item_path** (Optional[str]) – path to item to be removed (e.g. `Folder\Subfolder\ItemName`)
- **new_item_path** (Optional[str]) – path to item to be removed (e.g. `Folder\Subfolder\ItemName`)

**Return type**
None

**rss_refresh_item**(*item_path=None*, ***kwargs*) → None

Trigger a refresh for an RSS item.

Note: qBittorrent v4.1.5 through v4.1.8 all use Web API 2.2 but this endpoint was introduced with v4.1.8; so, behavior may be undefined for these versions.

**Parameters**
**item_path** (Optional[str]) – path to item to be refreshed (e.g. `Folder\Subfolder\ItemName`)

**Return type**
None

**rss_remove_item**(*item_path=None*, ***kwargs*) → None

Remove an RSS item (folder, feed, etc).

NOTE: Removing a folder also removes everything in it.

**Raises**
*Conflict409Error* –

**Parameters**
**item_path** (Optional[str]) – path to item to be removed (e.g. `Folder\Subfolder\ItemName`)

**Return type**
None

**rss_remove_rule**(*rule_name=None*, ***kwargs*) → None

Delete a RSS auto-downloading rule.

**Parameters**
**rule_name** (Optional[str]) – Name of rule to delete

**Return type**
None

**rss_rename_rule**(*orig_rule_name=None*, *new_rule_name=None*, ***kwargs*) → None

Rename an RSS auto-download rule.

This method did not work properly until qBittorrent v4.3.0 (Web API v2.6).

**Parameters**

- **orig_rule_name** (Optional[str]) – current name of rule
- **new_rule_name** (Optional[str]) – new name for rule

**Return type**
None

**rss_rules**(***kwargs*) → *RSSRulesDictionary*

Retrieve RSS auto-download rule definitions.

**Return type**
*RSSRulesDictionary*

**rss_set_feed_url**(*url=None*, *item_path=None*, ***kwargs*) → None

Update the URL for an existing RSS feed.

This method was introduced with qBittorrent v4.6.0 (Web API v2.9.1).

**Raises**
*Conflict409Error* –

**Parameters**

- **url** (Optional[str]) – URL of RSS feed (e.g. https://distrowatch.com/news/torrents.xml)
- **item_path** (Optional[str]) – Name and/or path for feed (e.g. `Folder\Subfolder\FeedName`)

**Return type**
None

**rss_set_rule**(*rule_name=None*, *rule_def=None*, ***kwargs*) → None

Create a new RSS auto-downloading rule.

**Parameters**

- **rule_name** (Optional[str]) – name for new rule
- **rule_def** (Optional[Mapping[str, Union[None, int, str, bool, Sequence[Union[None, int, str, bool, Sequence[JsonValueT], Mapping[str, JsonValueT]]], Mapping[str, Union[None, int, str, bool, Sequence[JsonValueT], Mapping[str, JsonValueT]]]]]]) – dictionary with rule fields - https://github.com/qbittorrent/qBittorrent/wiki/WebUI-API-(qBittorrent-4.1)#user-content-set-auto-downloading-rule

**Return type**
None

**class RSS**(*client*)

Allows interaction with RSS API endpoints.

**Usage**

```
>>> from qbittorrentapi import Client
>>> client = Client(host="localhost:8080", username="admin", password=
↪"adminadmin")
>>> # this is all the same attributes that are available as named in the
>>> #  endpoints or the more pythonic names in Client (with or without
↪'log_' prepended)
>>> rss_rules = client.rss.rules
>>> client.rss.addFolder(folder_path="TPB")
>>> client.rss.addFeed(url="...", item_path="TPB\Top100")
>>> client.rss.remove_item(item_path="TPB") # deletes TPB and Top100
>>> client.rss.set_rule(rule_name="...", rule_def={...})
>>> items = client.rss.items.with_data
>>> items_no_data = client.rss.items.without_data
```

**add_feed**(*url=None*, *item_path=None*, ***kwargs*) → None

Add new RSS feed. Folders in path must already exist.

**Raises**
*Conflict409Error* –

**Parameters**

- **url** (Optional[str]) – URL of RSS feed (e.g. https://distrowatch.com/news/torrents.xml)
- **item_path** (Optional[str]) – Name and/or path for new feed (e.g. `Folder\Subfolder\FeedName`)

**Return type**
None

**add_folder**(*folder_path=None*, ***kwargs*) → None

Add an RSS folder. Any intermediate folders in path must already exist.

**Raises**
*Conflict409Error* –

**Parameters**
**folder_path** (Optional[str]) – path to new folder (e.g. Linux\ISOs)

**Return type**
None

**mark_as_read**(*item_path=None*, *article_id=None*, ***kwargs*) → None

Mark RSS article as read. If article ID is not provider, the entire feed is marked as read.

This method was introduced with qBittorrent v4.2.5 (Web API v2.5.1).

**Raises**
*NotFound404Error* –

**Parameters**

- **item_path** (Optional[str]) – path to item to be refreshed (e.g. Folder\Subfolder\ItemName)
- **article_id** (UnionType[str, int, None]) – article ID from *rss_items()*

**Return type**
None

**matching_articles**(*rule_name=None*, ***kwargs*) → *RSSitemsDictionary*

Fetch all articles matching a rule.

This method was introduced with qBittorrent v4.2.5 (Web API v2.5.1).

**Parameters**
**rule_name** (Optional[str]) – Name of rule to return matching articles

**Return type**
*RSSitemsDictionary*

**move_item**(*orig_item_path=None*, *new_item_path=None*, ***kwargs*) → None

Move/rename an RSS item (folder, feed, etc.).

**Raises**
*Conflict409Error* –

**Parameters**

- **orig_item_path** (Optional[str]) – path to item to be removed (e.g. Folder\Subfolder\ItemName)
- **new_item_path** (Optional[str]) – path to item to be removed (e.g. Folder\Subfolder\ItemName)

**Return type**
None

**refresh_item**(*item_path=None*, ***kwargs*) → None

Trigger a refresh for an RSS item.

Note: qBittorrent v4.1.5 through v4.1.8 all use Web API 2.2 but this endpoint was introduced with v4.1.8; so, behavior may be undefined for these versions.

**Parameters**
**item_path** (Optional[str]) – path to item to be refreshed (e.g. Folder\Subfolder\ItemName)

**Return type**
None

**remove_item**(*item_path=None*, ***kwargs*) → None

Remove an RSS item (folder, feed, etc).

NOTE: Removing a folder also removes everything in it.

**Raises**
*Conflict409Error* –

**Parameters**
**item_path** (Optional[str]) – path to item to be removed (e.g. `Folder\Subfolder\ItemName`)

**Return type**
None

**remove_rule**(*rule_name=None*, ***kwargs*) → None

Delete a RSS auto-downloading rule.

**Parameters**
**rule_name** (Optional[str]) – Name of rule to delete

**Return type**
None

**rename_rule**(*orig_rule_name=None*, *new_rule_name=None*, ***kwargs*) → None

Rename an RSS auto-download rule.

This method did not work properly until qBittorrent v4.3.0 (Web API v2.6).

**Parameters**

- **orig_rule_name** (Optional[str]) – current name of rule
- **new_rule_name** (Optional[str]) – new name for rule

**Return type**
None

**property rules:** *RSSRulesDictionary*

Retrieve RSS auto-download rule definitions.

**set_feed_url**(*url=None*, *item_path=None*, ***kwargs*) → None

Update the URL for an existing RSS feed.

This method was introduced with qBittorrent v4.6.0 (Web API v2.9.1).

**Raises**
*Conflict409Error* –

**Parameters**

- **url** (Optional[str]) – URL of RSS feed (e.g. https://distrowatch.com/news/torrents.xml)
- **item_path** (Optional[str]) – Name and/or path for feed (e.g. `Folder\Subfolder\FeedName`)

**Return type**
None

**set_rule**(*rule_name=None*, *rule_def=None*, ***kwargs*) → None

Create a new RSS auto-downloading rule.

**Parameters**

- **rule_name** (Optional[str]) – name for new rule
- **rule_def** (Optional[Mapping[str, Union[None, int, str, bool, Sequence[Union[None, int, str, bool, Sequence[JsonValueT], Mapping[str, JsonValueT]]], Mapping[str, Union[None, int, str, bool, Sequence[JsonValueT], Mapping[str, JsonValueT]]]]]]) – dictionary with rule fields - https://github.com/qbittorrent/qBittorrent/wiki/WebUI-API-(qBittorrent-4.1)#user-content-set-auto-downloading-rule

**Return type**
None

**class RSSitemsDictionary**(*data=None*, ***kwargs*)

Bases: *Dictionary*[Union[None, int, str, bool, Sequence[JsonValueT], Mapping[str, JsonValueT]]]

Response for *rss_items()*

**class RSSRulesDictionary**(*data=None*, ***kwargs*)

Bases: *Dictionary*[Union[None, int, str, bool, Sequence[JsonValueT], Mapping[str, JsonValueT]]]

Response for *rss_rules()*

## Search

**class SearchAPIMixIn**(*host=None*, *port=None*, *username=None*, *password=None*, *EXTRA_HEADERS=None*, *REQUESTS_ARGS=None*, *VERIFY_WEBUI_CERTIFICATE=True*, *FORCE_SCHEME_FROM_HOST=False*, *RAISE_NOTIMPLEMENTEDERROR_FOR_UNIMPLEMENTED_API_ENDPOINTS=False*, *RAISE_ERROR_FOR_UNSUPPORTED_QBITTORRENT_VERSIONS=False*, *VERBOSE_RESPONSE_LOGGING=False*, *SIMPLE_RESPONSES=False*, *DISABLE_LOGGING_DEBUG_OUTPUT=False*) → None

Bases: *AppAPIMixIn*

Implementation for all Search API methods.

**Usage**

```
>>> from qbittorrentapi import Client
>>> client = Client(host="localhost:8080", username="admin", password=
↪"adminadmin")
>>> search_job = client.search_start(pattern="Ubuntu", plugins="all",␣
↪category="all")
>>> client.search_stop(search_id=search_job.id)
>>> # or
>>> search_job.stop()
```

**search_categories**(*plugin_name=None*, ***kwargs*) → *SearchCategoriesList*

Retrieve categories for search.

This method was introduced with qBittorrent v4.1.4 (Web API v2.1.1) and removed with qBittorrent v4.3.0 (Web API v2.6).

**Parameters**
**plugin_name** (Optional[str]) – Limit categories returned by plugin(s) (supports `all` and `enabled`)

**Return type**
*SearchCategoriesList*

**search_delete**(*search_id=None*, ***kwargs*) → None

Delete a search job.

This method was introduced with qBittorrent v4.1.4 (Web API v2.1.1).

**Raises**
***NotFound404Error*** –

**Parameters**
**search_id** (UnionType[str, int, None]) – ID of search to delete

**Return type**
None

**search_enable_plugin**(*plugins=None*, *enable=None*, ***kwargs*) → None

Enable or disable search plugin(s).

This method was introduced with qBittorrent v4.1.4 (Web API v2.1.1).

**Parameters**

- **plugins** (Optional[Iterable[str]]) – list of plugin names
- **enable** (Optional[bool]) – Defaults to True if None or unset; use False to disable

**Return type**
None

**search_install_plugin**(*sources=None*, ***kwargs*) → None

Install search plugins from either URL or file.

This method was introduced with qBittorrent v4.1.4 (Web API v2.1.1).

**Parameters**
**sources** (Optional[Iterable[str]]) – list of URLs or filepaths

**Return type**
None

**search_plugins**(***kwargs*) → *SearchPluginsList*

Retrieve details of search plugins.

This method was introduced with qBittorrent v4.1.4 (Web API v2.1.1).

**Return type**
*SearchPluginsList*

**search_results**(*search_id=None*, *limit=None*, *offset=None*, ***kwargs*) → *SearchResultsDictionary*

Retrieve the results for the search.

This method was introduced with qBittorrent v4.1.4 (Web API v2.1.1).

**Raises**

- ***NotFound404Error*** –
- ***Conflict409Error*** –

**Parameters**

- **search_id** (UnionType[str, int, None]) – ID of search job
- **limit** (UnionType[str, int, None]) – number of results to return
- **offset** (UnionType[str, int, None]) – where to start returning results

**Return type**
*SearchResultsDictionary*

**search_start**(*pattern=None*, *plugins=None*, *category=None*, ***kwargs*) → *SearchJobDictionary*

Start a search. Python must be installed. Host may limit number of concurrent searches.

This method was introduced with qBittorrent v4.1.4 (Web API v2.1.1).

**Raises**
***Conflict409Error*** –

**Parameters**

- **pattern** (Optional[str]) – term to search for
- **plugins** (Optional[Iterable[str]]) – list of plugins to use for searching (supports 'all' and 'enabled')
- **category** (Optional[str]) – categories to limit search; dependent on plugins. (supports 'all')

**Return type**
*SearchJobDictionary*

**search_status**(*search_id=None*, ***kwargs*) → *SearchStatusesList*

Retrieve status of one or all searches.

This method was introduced with qBittorrent v4.1.4 (Web API v2.1.1).

**Raises**
***NotFound404Error*** –

**Parameters**
**search_id** (UnionType[str, int, None]) – ID of search to get status; leave empty for status of all jobs

**Return type**
*SearchStatusesList*

**search_stop**(*search_id=None*, ***kwargs*) → None

Stop a running search.

This method was introduced with qBittorrent v4.1.4 (Web API v2.1.1).

**Raises**
***NotFound404Error*** –

**Parameters**
**search_id** (UnionType[str, int, None]) – ID of search job to stop

**Return type**
None

**search_uninstall_plugin**(*names=None*, ***kwargs*) → None

Uninstall search plugins.

This method was introduced with qBittorrent v4.1.4 (Web API v2.1.1).

**Parameters**
**names** (Optional[Iterable[str]]) – names of plugins to uninstall

**Return type**
None

**search_update_plugins**(***kwargs*) → None

Auto update search plugins.

This method was introduced with qBittorrent v4.1.4 (Web API v2.1.1).

**Return type**
None

**class Search**(**args*, *client*, ***kwargs*)

Allows interaction with Search API endpoints.

**Usage**

```
>>> from qbittorrentapi import Client
>>> client = Client(host="localhost:8080", username="admin", password=
↪"adminadmin")
>>> # this is all the same attributes that are available as named in the
>>> #  endpoints or the more pythonic names in Client (with or without
↪'search_' prepended)
>>> # initiate searches and retrieve results
>>> search_job = client.search.start(pattern="Ubuntu", plugins="all",␣
↪category="all")
>>> status = search_job.status()
>>> results = search_job.result()
>>> search_job.delete()
>>> # inspect and manage plugins
>>> plugins = client.search.plugins
>>> cats = client.search.categories(plugin_name="...")
>>> client.search.install_plugin(sources="...")
>>> client.search.update_plugins()
```

**categories**(*plugin_name=None*, ***kwargs*) → *SearchCategoriesList*

Retrieve categories for search.

This method was introduced with qBittorrent v4.1.4 (Web API v2.1.1) and removed with qBittorrent v4.3.0 (Web API v2.6).

**Parameters**
**plugin_name** (Optional[str]) – Limit categories returned by plugin(s) (supports all and enabled)

**Return type**
*SearchCategoriesList*

**delete**(*search_id=None*, ***kwargs*) → None

Delete a search job.

This method was introduced with qBittorrent v4.1.4 (Web API v2.1.1).

**Raises**
***NotFound404Error*** –

**Parameters**
**search_id** (UnionType[str, int, None]) – ID of search to delete

**Return type**
None

**enable_plugin**(*plugins=None*, *enable=None*, ***kwargs*) → None

Enable or disable search plugin(s).

This method was introduced with qBittorrent v4.1.4 (Web API v2.1.1).

**Parameters**

- **plugins** (Optional[Iterable[str]]) – list of plugin names
- **enable** (Optional[bool]) – Defaults to `True` if `None` or unset; use `False` to disable

**Return type**
None

**install_plugin**(*sources=None*, ***kwargs*) → None

Install search plugins from either URL or file.

This method was introduced with qBittorrent v4.1.4 (Web API v2.1.1).

**Parameters**
**sources** (Optional[Iterable[str]]) – list of URLs or filepaths

**Return type**
None

**property plugins:** *SearchPluginsList*

Retrieve details of search plugins.

This method was introduced with qBittorrent v4.1.4 (Web API v2.1.1).

**results**(*search_id=None*, *limit=None*, *offset=None*, ***kwargs*) → *SearchResultsDictionary*

Retrieve the results for the search.

This method was introduced with qBittorrent v4.1.4 (Web API v2.1.1).

**Raises**

- *NotFound404Error* –
- *Conflict409Error* –

**Parameters**

- **search_id** (UnionType[str, int, None]) – ID of search job
- **limit** (UnionType[str, int, None]) – number of results to return
- **offset** (UnionType[str, int, None]) – where to start returning results

**Return type**
*SearchResultsDictionary*

**start**(*pattern=None*, *plugins=None*, *category=None*, ***kwargs*) → *SearchJobDictionary*

Start a search. Python must be installed. Host may limit number of concurrent searches.

This method was introduced with qBittorrent v4.1.4 (Web API v2.1.1).

**Raises**
*Conflict409Error* –

**Parameters**

- **pattern** (Optional[str]) – term to search for
- **plugins** (Optional[Iterable[str]]) – list of plugins to use for searching (supports 'all' and 'enabled')

- **category** (Optional[str]) – categories to limit search; dependent on plugins. (supports 'all')

**Return type**
*SearchJobDictionary*

**status**(*search_id=None*, ***kwargs*) → *SearchStatusesList*

Retrieve status of one or all searches.

This method was introduced with qBittorrent v4.1.4 (Web API v2.1.1).

**Raises**
*NotFound404Error* –

**Parameters**
**search_id** (UnionType[str, int, None]) – ID of search to get status; leave empty for status of all jobs

**Return type**
*SearchStatusesList*

**stop**(*search_id=None*, ***kwargs*) → None

Stop a running search.

This method was introduced with qBittorrent v4.1.4 (Web API v2.1.1).

**Raises**
*NotFound404Error* –

**Parameters**
**search_id** (UnionType[str, int, None]) – ID of search job to stop

**Return type**
None

**uninstall_plugin**(*names=None*, ***kwargs*) → None

Uninstall search plugins.

This method was introduced with qBittorrent v4.1.4 (Web API v2.1.1).

**Parameters**
**names** (Optional[Iterable[str]]) – names of plugins to uninstall

**Return type**
None

**update_plugins**(***kwargs*) → None

Auto update search plugins.

This method was introduced with qBittorrent v4.1.4 (Web API v2.1.1).

**Return type**
None

**class SearchJobDictionary**(*data*, *client*)

Bases: *ClientCache*[*SearchAPIMixIn*], *Dictionary*[Union[None, int, str, bool, Sequence[JsonValueT], Mapping[str, JsonValueT]]]

Response for *search_start()*

**delete**(*search_id=None*, ***kwargs*) → None

Delete a search job.

This method was introduced with qBittorrent v4.1.4 (Web API v2.1.1).

**Raises**
*NotFound404Error* –

**Parameters**
**search_id** (UnionType[str, int, None]) – ID of search to delete

**Return type**
None

**results**(*search_id=None*, *limit=None*, *offset=None*, ***kwargs*) → *SearchResultsDictionary*

Retrieve the results for the search.

This method was introduced with qBittorrent v4.1.4 (Web API v2.1.1).

**Raises**
- *NotFound404Error* –
- *Conflict409Error* –

**Parameters**
- **search_id** (UnionType[str, int, None]) – ID of search job
- **limit** (UnionType[str, int, None]) – number of results to return
- **offset** (UnionType[str, int, None]) – where to start returning results

**Return type**
*SearchResultsDictionary*

**status**(*search_id=None*, ***kwargs*) → *SearchStatusesList*

Retrieve status of one or all searches.

This method was introduced with qBittorrent v4.1.4 (Web API v2.1.1).

**Raises**
*NotFound404Error* –

**Parameters**
**search_id** (UnionType[str, int, None]) – ID of search to get status; leave empty for status of all jobs

**Return type**
*SearchStatusesList*

**stop**(*search_id=None*, ***kwargs*) → None

Stop a running search.

This method was introduced with qBittorrent v4.1.4 (Web API v2.1.1).

**Raises**
*NotFound404Error* –

**Parameters**
**search_id** (UnionType[str, int, None]) – ID of search job to stop

**Return type**
None

**class SearchResultsDictionary**(*data=None*, ***kwargs*)

Bases: *Dictionary*[Union[None, int, str, bool, Sequence[JsonValueT], Mapping[str, JsonValueT]]]

Response for *search_results()*

Definition: https://github.com/qbittorrent/qBittorrent/wiki/WebUI-API-(qBittorrent-4.1)#user-content-get-search-results

**class SearchStatusesList**(*list_entries*, *client=None*)

Bases: *List*[*SearchStatus*]

Response for *search_status()*

Definition: https://github.com/qbittorrent/qBittorrent/wiki/WebUI-API-(qBittorrent-4.1)#user-content-get-search-status

**class SearchStatus**(*data=None*, ***kwargs*)

Bases: *ListEntry*

Item in *SearchStatusesList*

**class SearchCategoriesList**(*list_entries*, *client=None*)

Bases: *List*[*SearchCategory*]

Response for *search_categories()*

**class SearchCategory**(*data=None*, ***kwargs*)

Bases: *ListEntry*

Item in *SearchCategoriesList*

**class SearchPluginsList**(*list_entries*, *client=None*)

Bases: *List*[*SearchPlugin*]

Response for *search_plugins()*.

Definition: https://github.com/qbittorrent/qBittorrent/wiki/WebUI-API-(qBittorrent-4.1)#user-content-get-search-plugins

**class SearchPlugin**(*data=None*, ***kwargs*)

Bases: *ListEntry*

Item in *SearchPluginsList*

### Sync

**class SyncAPIMixIn**(*host=None*, *port=None*, *username=None*, *password=None*, *EXTRA_HEADERS=None*, *REQUESTS_ARGS=None*, *VERIFY_WEBUI_CERTIFICATE=True*, *FORCE_SCHEME_FROM_HOST=False*, *RAISE_NOTIMPLEMENTEDERROR_FOR_UNIMPLEMENTED_API_ENDPOINTS=False*, *RAISE_ERROR_FOR_UNSUPPORTED_QBITTORRENT_VERSIONS=False*, *VERBOSE_RESPONSE_LOGGING=False*, *SIMPLE_RESPONSES=False*, *DISABLE_LOGGING_DEBUG_OUTPUT=False*) → None

Bases: *AppAPIMixIn*

Implementation of all Sync API Methods.

**Usage**

```
>>> from qbittorrentapi import Client
>>> client = Client(host="localhost:8080", username="admin", password=
↪"adminadmin")
>>> maindata = client.sync_maindata(rid="...")
>>> torrent_peers = client.sync_torrent_peers(torrent_hash="...", rid="..
↪..")
```

**sync_maindata**(*rid=0*, ***kwargs*) → *SyncMainDataDictionary*

Retrieves sync data.

**Parameters**
**rid** (str | int) – response ID

**Return type**
*SyncMainDataDictionary*

**sync_torrent_peers**(*torrent_hash=None*, *rid=0*, ***kwargs*) → *SyncTorrentPeersDictionary*

Retrieves torrent sync data.

**Raises**
***NotFound404Error*** –

**Parameters**

- **torrent_hash** (Optional[str]) – hash for torrent
- **rid** (str | int) – response ID

**Return type**
*SyncTorrentPeersDictionary*

**class Sync**(*client*) → None

Allows interaction with the Sync API endpoints.

**Usage:**

```
>>> from qbittorrentapi import Client
>>> client = Client(host="localhost:8080", username="admin", password=
↪"adminadmin")
>>> # these are all the same attributes that are available as named in the
>>> #  endpoints or the more pythonic names in Client (with or without 'sync_'␣
↪prepended)
>>> maindata = client.sync.maindata(rid="...")
>>> # for use when continuously calling maindata for changes in torrents
>>> # this will automatically request the changes since the last call
>>> md = client.sync.maindata.delta()
>>> #
>>> torrentPeers = client.sync.torrentPeers(torrent_hash="...", rid="...")
>>> torrent_peers = client.sync.torrent_peers(torrent_hash="...", rid="...")
```

**class SyncMainDataDictionary**(*data=None*, ***kwargs*)

Bases: *Dictionary*[Union[None, int, str, bool, Sequence[JsonValueT], Mapping[str, JsonValueT]]]

Response for *sync_maindata()*

Definition: https://github.com/qbittorrent/qBittorrent/wiki/WebUI-API-(qBittorrent-4.1)#user-content-get-main-data

**class SyncTorrentPeersDictionary**(*data=None*, ***kwargs*)

Bases: *Dictionary*[Union[None, int, str, bool, Sequence[JsonValueT], Mapping[str, JsonValueT]]]

Response for *sync_torrent_peers()*

Definition: https://github.com/qbittorrent/qBittorrent/wiki/WebUI-API-(qBittorrent-4.1)#user-content-get-torrent-peers-data

### Torrents

**class TorrentsAPIMixIn**(*host=None*, *port=None*, *username=None*, *password=None*, *EXTRA_HEADERS=None*, *REQUESTS_ARGS=None*, *VERIFY_WEBUI_CERTIFICATE=True*, *FORCE_SCHEME_FROM_HOST=False*, *RAISE_NOTIMPLEMENTEDERROR_FOR_UNIMPLEMENTED_API_ENDPOINTS=False*, *RAISE_ERROR_FOR_UNSUPPORTED_QBITTORRENT_VERSIONS=False*, *VERBOSE_RESPONSE_LOGGING=False*, *SIMPLE_RESPONSES=False*, *DISABLE_LOGGING_DEBUG_OUTPUT=False*) → None

Bases: *AppAPIMixIn*

Implementation of all Torrents API methods.

**Usage**

```
>>> from qbittorrentapi import Client
>>> client = Client(host="localhost:8080", username="admin", password=
↪"adminadmin")
>>> client.torrents_add(urls="...")
>>> client.torrents_reannounce()
```

**torrents_add**(*urls=None*, *torrent_files=None*, *save_path=None*, *cookie=None*, *category=None*, *is_skip_checking=None*, *is_paused=None*, *is_root_folder=None*, *rename=None*, *upload_limit=None*, *download_limit=None*, *use_auto_torrent_management=None*, *is_sequential_download=None*, *is_first_last_piece_priority=None*, *tags=None*, *content_layout=None*, *ratio_limit=None*, *seeding_time_limit=None*, *download_path=None*, *use_download_path=None*, *stop_condition=None*, ***kwargs*) → str

Add one or more torrents by URLs and/or torrent files.

Returns `Ok.` for success and `Fails.` for failure.

**Raises**

- *UnsupportedMediaType415Error* – if file is not a valid torrent file
- *TorrentFileNotFoundError* – if a torrent file doesn't exist
- *TorrentFilePermissionError* – if read permission is denied to torrent file

**Parameters**

- **urls** (Optional[Iterable[str]]) – single instance or an iterable of URLs (`http://`, `https://`, `magnet:`, `bc://bt/`)
- **torrent_files** (Optional[TypeVar(TorrentFilesT, bytes, str, IO[bytes], Mapping[str, Union[bytes, str, IO[bytes]]], Iterable[Union[bytes, str, IO[bytes]]])]) – several options are available to send torrent files to qBittorrent:
  - single instance of bytes: useful if torrent file already read from disk or downloaded from internet.
  - single instance of file handle to torrent file: use `open(<filepath>, 'rb')` to open the torrent file.
  - single instance of a filepath to torrent file: e.g. `/home/user/torrent_filename.torrent`
  - an iterable of the single instances above to send more than one torrent file
  - dictionary with key/value pairs of torrent name and single instance of above object

Note: The torrent name in a dictionary is useful to identify which torrent file errored. qBittorrent provides back that name in the error text. If a torrent name is not provided, then the name of the file will be used. And in the case of bytes (or if filename cannot be determined), the value 'torrent__n' will be used.

- **save_path** (Optional[str]) – location to save the torrent data
- **cookie** (Optional[str]) – cookie to retrieve torrents by URL
- **category** (Optional[str]) – category to assign to torrent(s)
- **is_skip_checking** (Optional[bool]) – True to skip hash checking
- **is_paused** (Optional[bool]) – True to add the torrent(s) without starting their downloading
- **is_root_folder** (Optional[bool]) – True or False to create root folder (superseded by content_layout with v4.3.2)
- **rename** (Optional[str]) – new name for torrent(s)
- **upload_limit** (UnionType[str, int, None]) – upload limit in bytes/second
- **download_limit** (UnionType[str, int, None]) – download limit in bytes/second
- **use_auto_torrent_management** (Optional[bool]) – True or False to use automatic torrent management
- **is_sequential_download** (Optional[bool]) – True or False for sequential download
- **is_first_last_piece_priority** (Optional[bool]) – True or False for first and last piece download priority
- **tags** (Optional[Iterable[str]]) – tag(s) to assign to torrent(s) (added in Web API 2.6.2)
- **content_layout** (Optional[Literal['Original', 'Subfolder', 'NoSubFolder']]) – Original, Subfolder, or NoSubfolder to control filesystem structure for content (added in Web API 2.7)
- **ratio_limit** (UnionType[str, float, None]) – share limit as ratio of upload amt over download amt; e.g. 0.5 or 2.0 (added in Web API 2.8.1)
- **seeding_time_limit** (UnionType[str, int, None]) – number of minutes to seed torrent (added in Web API 2.8.1)
- **download_path** (Optional[str]) – location to download torrent content before moving to save_path (added in Web API 2.8.4)
- **use_download_path** (Optional[bool]) – True or False whether download_path should be used. . . defaults to True if download_path is specified (added in Web API 2.8.4)
- **stop_condition** (Optional[Literal['MetadataReceived', 'FilesChecked']]) – MetadataReceived or FilesChecked to stop the torrent when started automatically (added in Web API 2.8.15)

**Return type**
str

**torrents_add_peers**(*peers=None*, *torrent_hashes=None*, ***kwargs*) → *TorrentsAddPeersDictionary*

Add one or more peers to one or more torrents.

This method was introduced with qBittorrent v4.4.0 (Web API v2.3.0).

**Raises**
*InvalidRequest400Error* – for invalid peers

**Parameters**

- **peers** (Optional[Iterable[str]]) – one or more peers to add. each peer should take the form 'host:port'
- **torrent_hashes** (Optional[Iterable[str]]) – single torrent hash or list of torrent hashes. Or `all` for all torrents.

**Return type**
*TorrentsAddPeersDictionary*

**torrents_add_tags**(*tags=None*, *torrent_hashes=None*, ***kwargs*) → None

Add one or more tags to one or more torrents.

Note: Tags that do not exist will be created on-the-fly.

This method was introduced with qBittorrent v4.2.0 (Web API v2.3.0).

**Parameters**

- **tags** (Optional[Iterable[str]]) – tag name or list of tags
- **torrent_hashes** (Optional[Iterable[str]]) – single torrent hash or list of torrent hashes. Or `all` for all torrents.

**Return type**
None

**torrents_add_trackers**(*torrent_hash=None*, *urls=None*, ***kwargs*) → None

Add trackers to a torrent.

**Raises**
*NotFound404Error* –

**Parameters**

- **torrent_hash** (Optional[str]) – hash for torrent
- **urls** (Optional[Iterable[str]]) – tracker URLs to add to torrent

**Return type**
None

**torrents_bottom_priority**(*torrent_hashes=None*, ***kwargs*) → None

Set torrent as lowest priority. Torrent Queuing must be enabled.

**Raises**
*Conflict409Error* –

**Parameters**
**torrent_hashes** (Optional[Iterable[str]]) – single torrent hash or list of torrent hashes. Or `all` for all torrents.

**Return type**
None

**torrents_categories**(***kwargs*) → *TorrentCategoriesDictionary*

Retrieve all category definitions.

This method was introduced with qBittorrent v4.1.4 (Web API v2.1.1).

Note: `torrents/categories` is not available until v2.1.0

**Return type**
*TorrentCategoriesDictionary*

**torrents_count**() → int

Retrieve count of torrents.

**Return type**
int

**torrents_create_category**(*name=None*, *save_path=None*, *download_path=None*, *enable_download_path=None*, ***kwargs*) → None

Create a new torrent category.

**Raises**
***Conflict409Error*** – if category name is not valid or unable to create

**Parameters**

- **name** (Optional[str]) – name for new category
- **save_path** (Optional[str]) – location to save torrents for this category (added in Web API 2.1.0)
- **download_path** (Optional[str]) – download location for torrents with this category
- **enable_download_path** (Optional[bool]) – True or False to enable or disable download path

**Return type**
None

**torrents_create_tags**(*tags=None*, ***kwargs*) → None

Create one or more tags.

This method was introduced with qBittorrent v4.2.0 (Web API v2.3.0).

**Parameters**
**tags** (Optional[Iterable[str]]) – tag name or list of tags

**Return type**
None

**torrents_decrease_priority**(*torrent_hashes=None*, ***kwargs*) → None

Decrease the priority of a torrent. Torrent Queuing must be enabled.

**Raises**
***Conflict409Error*** –

**Parameters**
**torrent_hashes** (Optional[Iterable[str]]) – single torrent hash or list of torrent hashes. Or `all` for all torrents.

**Return type**
None

**torrents_delete**(*delete_files=False*, *torrent_hashes=None*, ***kwargs*) → None

Remove a torrent from qBittorrent and optionally delete its files.

**Parameters**

- **torrent_hashes** (Optional[Iterable[str]]) – single torrent hash or list of torrent hashes. Or `all` for all torrents.
- **delete_files** (Optional[bool]) – True to delete the torrent's files

**Return type**
None

**torrents_delete_tags**(*tags=None*, ***kwargs*) → None

Delete one or more tags.

This method was introduced with qBittorrent v4.2.0 (Web API v2.3.0).

**Parameters**
**tags** (Optional[Iterable[str]]) – tag name or list of tags

**Return type**
None

**torrents_download_limit**(*torrent_hashes=None*, ***kwargs*) → *TorrentLimitsDictionary*

Retrieve the download limit for one or more torrents.

**Return type**
*TorrentLimitsDictionary*

**torrents_edit_category**(*name=None*, *save_path=None*, *download_path=None*, *enable_download_path=None*, ***kwargs*) → None

Edit an existing category.

This method was introduced with qBittorrent v4.1.3 (Web API v2.1.0).

**Raises**
***Conflict409Error*** – if category name is not valid or unable to create

**Parameters**

- **name** (Optional[str]) – category to edit
- **save_path** (Optional[str]) – new location to save files for this category
- **download_path** (Optional[str]) – download location for torrents with this category
- **enable_download_path** (Optional[bool]) – True or False to enable or disable download path

**Return type**
None

**torrents_edit_tracker**(*torrent_hash=None*, *original_url=None*, *new_url=None*, ***kwargs*) → None

Replace a torrent's tracker with a different one.

This method was introduced with qBittorrent v4.1.4 (Web API v2.2.0).

**Raises**

- ***InvalidRequest400Error*** –
- ***NotFound404Error*** –
- ***Conflict409Error*** –

**Parameters**

- **torrent_hash** (Optional[str]) – hash for torrent
- **original_url** (Optional[str]) – URL for existing tracker
- **new_url** (Optional[str]) – new URL to replace

**Return type**
None

**torrents_export**(*torrent_hash=None*, ***kwargs*) → bytes

Export a .torrent file for the torrent.

This method was introduced with qBittorrent v4.5.0 (Web API v2.8.14).

**Raises**

- ***NotFound404Error*** – torrent not found
- ***Conflict409Error*** – unable to export .torrent file

**Parameters**

**torrent_hash** (Optional[str]) – hash for torrent

**Return type**

bytes

**torrents_file_priority**(*torrent_hash=None*, *file_ids=None*, *priority=None*, ***kwargs*) → None

Set priority for one or more files.

**Raises**

- ***InvalidRequest400Error*** – if priority is invalid or at least one file ID is not an integer
- ***NotFound404Error*** –
- ***Conflict409Error*** – if torrent metadata has not finished downloading or at least one file was not found

**Parameters**

- **torrent_hash** (Optional[str]) – hash for torrent
- **file_ids** (Union[int, Iterable[str | int], None]) – single file ID or a list.
- **priority** (UnionType[str, int, None]) – priority for file(s) - https://github.com/qbittorrent/qBittorrent/wiki/WebUI-API-(qBittorrent-4.1)#user-content-set-file-priority

**Return type**

None

**torrents_files**(*torrent_hash=None*, ***kwargs*) → *TorrentFilesList*

Retrieve individual torrent's files.

**Raises**

***NotFound404Error*** –

**Parameters**

**torrent_hash** (Optional[str]) – hash for torrent

**Return type**

*TorrentFilesList*

**torrents_increase_priority**(*torrent_hashes=None*, ***kwargs*) → None

Increase the priority of a torrent. Torrent Queuing must be enabled.

**Raises**

***Conflict409Error*** –

**Parameters**

**torrent_hashes** (Optional[Iterable[str]]) – single torrent hash or list of torrent hashes. Or all for all torrents.

**Return type**

None

**torrents_info**(*status_filter=None*, *category=None*, *sort=None*, *reverse=None*, *limit=None*, *offset=None*, *torrent_hashes=None*, *tag=None*, ***kwargs*) → *TorrentInfoList*

Retrieves list of info for torrents.

**Parameters**

- **status_filter** (Optional[Literal['all', 'downloading', 'seeding', 'completed', 'paused', 'active', 'inactive', 'resumed', 'stalled', 'stalled_uploading', 'stalled_downloading', 'checking', 'moving', 'errored']]) – Filter list by torrent status. `all`, `downloading`, `seeding`, `completed`, `paused` `active`, `inactive`, `resumed`, `errored` Added in Web API 2.4.1: `stalled`, `stalled_uploading`, and `stalled_downloading` Added in Web API 2.8.4: `checking` Added in Web API 2.8.18: `moving`
- **category** (Optional[str]) – Filter list by category
- **sort** (Optional[str]) – Sort list by any property returned
- **reverse** (Optional[bool]) – Reverse sorting
- **limit** (UnionType[str, int, None]) – Limit length of list
- **offset** (UnionType[str, int, None]) – Start of list (if < 0, offset from end of list)
- **torrent_hashes** (Optional[Iterable[str]]) – Filter list by hash (separate multiple hashes with a '|') (added in Web API 2.0.1)
- **tag** (Optional[str]) – Filter list by tag (empty string means "untagged"; no "tag" parameter means "any tag"; added in Web API 2.8.3)

**Return type**
*TorrentInfoList*

**torrents_pause**(*torrent_hashes=None*, ***kwargs*) → None

Pause one or more torrents in qBittorrent.

**Parameters**
**torrent_hashes** (Optional[Iterable[str]]) – single torrent hash or list of torrent hashes. Or `all` for all torrents.

**Return type**
None

**torrents_piece_hashes**(*torrent_hash=None*, ***kwargs*) → *TorrentPieceInfoList*

Retrieve individual torrent's pieces' hashes.

**Raises**
***NotFound404Error*** –

**Parameters**
**torrent_hash** (Optional[str]) – hash for torrent

**Return type**
*TorrentPieceInfoList*

**torrents_piece_states**(*torrent_hash=None*, ***kwargs*) → *TorrentPieceInfoList*

Retrieve individual torrent's pieces' states.

**Raises**
***NotFound404Error*** –

**Parameters**
**torrent_hash** (Optional[str]) – hash for torrent

**Return type**
*TorrentPieceInfoList*

**torrents_properties**(*torrent_hash=None*, ***kwargs*) → *TorrentPropertiesDictionary*

Retrieve individual torrent's properties.

**Raises**
***NotFound404Error*** –

**Parameters**
**torrent_hash** (Optional[str]) – hash for torrent

**Return type**
*TorrentPropertiesDictionary*

**torrents_reannounce**(*torrent_hashes=None*, ***kwargs*) → None

Reannounce a torrent.

This method was introduced with qBittorrent v4.1.2 (Web API v2.0.2).

**Parameters**
**torrent_hashes** (Optional[Iterable[str]]) – single torrent hash or list of torrent hashes. Or `all` for all torrents.

**Return type**
None

**torrents_recheck**(*torrent_hashes=None*, ***kwargs*) → None

Recheck a torrent in qBittorrent.

**Parameters**
**torrent_hashes** (Optional[Iterable[str]]) – single torrent hash or list of torrent hashes. Or `all` for all torrents.

**Return type**
None

**torrents_remove_categories**(*categories=None*, ***kwargs*) → None

Delete one or more categories.

**Parameters**
**categories** (Optional[Iterable[str]]) – categories to delete

**Return type**
None

**torrents_remove_tags**(*tags=None*, *torrent_hashes=None*, ***kwargs*) → None

Add one or more tags to one or more torrents.

This method was introduced with qBittorrent v4.2.0 (Web API v2.3.0).

**Parameters**

- **tags** (Optional[Iterable[str]]) – tag name or list of tags
- **torrent_hashes** (Optional[Iterable[str]]) – single torrent hash or list of torrent hashes. Or `all` for all torrents.

**Return type**
None

**torrents_remove_trackers**(*torrent_hash=None*, *urls=None*, ***kwargs*) → None

Remove trackers from a torrent.

This method was introduced with qBittorrent v4.1.4 (Web API v2.2.0).

**Raises**

- ***NotFound404Error*** –
- ***Conflict409Error*** –

**Parameters**

- **torrent_hash** (Optional[str]) – hash for torrent
- **urls** (Optional[Iterable[str]]) – tracker URLs to removed from torrent

**Return type**
None

**torrents_rename**(*torrent_hash=None*, *new_torrent_name=None*, ***kwargs*) → None

Rename a torrent.

**Raises**
***NotFound404Error*** –

**Parameters**

- **torrent_hash** (Optional[str]) – hash for torrent
- **new_torrent_name** (Optional[str]) – new name for torrent

**Return type**
None

**torrents_rename_file**(*torrent_hash=None*, *file_id=None*, *new_file_name=None*, *old_path=None*, *new_path=None*, ***kwargs*) → None

Rename a torrent file.

This method was introduced with qBittorrent v4.2.1 (Web API v2.4.0).

**Raises**

- ***MissingRequiredParameters400Error*** –
- ***NotFound404Error*** –
- ***Conflict409Error*** –

**Parameters**

- **torrent_hash** (Optional[str]) – hash for torrent
- **file_id** (UnionType[str, int, None]) – id for file (removed in Web API 2.7)
- **new_file_name** (Optional[str]) – new name for file (removed in Web API 2.7)
- **old_path** (Optional[str]) – path of file to rename (added in Web API 2.7)
- **new_path** (Optional[str]) – new path of file to rename (added in Web API 2.7)

**Return type**
None

**torrents_rename_folder**(*torrent_hash=None*, *old_path=None*, *new_path=None*, ***kwargs*) → None

Rename a torrent folder.

This method was introduced with qBittorrent v4.3.2 (Web API v2.7).

**Raises**

- ***MissingRequiredParameters400Error*** –
- ***NotFound404Error*** –
- ***Conflict409Error*** –

**Parameters**

- **torrent_hash** (Optional[str]) – hash for torrent
- **old_path** (Optional[str]) – path of file to rename (added in Web API 2.7)
- **new_path** (Optional[str]) – new path of file to rename (added in Web API 2.7)

**Return type**
None

**torrents_resume**(*torrent_hashes=None*, ***kwargs*) → None

Resume one or more torrents in qBittorrent.

**Parameters**
**torrent_hashes** (Optional[Iterable[str]]) – single torrent hash or list of torrent hashes. Or `all` for all torrents.

**Return type**
None

**torrents_set_auto_management**(*enable=None*, *torrent_hashes=None*, ***kwargs*) → None

Enable or disable automatic torrent management for one or more torrents.

**Parameters**

- **torrent_hashes** (Optional[Iterable[str]]) – single torrent hash or list of torrent hashes. Or `all` for all torrents.
- **enable** (Optional[bool]) – Defaults to `True` if `None` or unset; use `False` to disable

**Return type**
None

**torrents_set_category**(*category=None*, *torrent_hashes=None*, ***kwargs*) → None

Set a category for one or more torrents.

**Raises**
***Conflict409Error*** – for bad category

**Parameters**

- **torrent_hashes** (Optional[Iterable[str]]) – single torrent hash or list of torrent hashes. Or `all` for all torrents.
- **category** (Optional[str]) – category to assign to torrent

**Return type**
None

**torrents_set_download_limit**(*limit=None*, *torrent_hashes=None*, ***kwargs*) → None

Set the download limit for one or more torrents.

**Parameters**

- **torrent_hashes** (Optional[Iterable[str]]) – single torrent hash or list of torrent hashes. Or `all` for all torrents.
- **limit** (UnionType[str, int, None]) – bytes/second (-1 sets the limit to infinity)

**Return type**
None

**torrents_set_download_path**(*download_path=None*, *torrent_hashes=None*, ***kwargs*) → None

Set the Download Path for one or more torrents.

This method was introduced with qBittorrent v4.4.0 (Web API v2.8.4).

**Raises**

- ***Forbidden403Error*** – cannot write to directory
- ***Conflict409Error*** – cannot create directory

**Parameters**

- **download_path** (Optional[str]) – file path to save torrent contents before torrent finishes downloading
- **torrent_hashes** (Optional[Iterable[str]]) – single torrent hash or list of torrent hashes. Or `all` for all torrents.

**Return type**
None

**torrents_set_force_start**(*enable=None*, *torrent_hashes=None*, ***kwargs*) → None

Force start one or more torrents.

**Parameters**

- **torrent_hashes** (Optional[Iterable[str]]) – single torrent hash or list of torrent hashes. Or `all` for all torrents.
- **enable** (Optional[bool]) – Defaults to `True` if `None` or unset; `False` is equivalent to *torrents_resume()*.

**Return type**
None

**torrents_set_location**(*location=None*, *torrent_hashes=None*, ***kwargs*) → None

Set location for torrents' files.

**Raises**

- ***Forbidden403Error*** – if the user doesn't have permissions to write to the location (only before v4.5.2 - write check was removed.)
- ***Conflict409Error*** – if the directory cannot be created at the location

**Parameters**

- **torrent_hashes** (Optional[Iterable[str]]) – single torrent hash or list of torrent hashes. Or `all` for all torrents.
- **location** (Optional[str]) – disk location to move torrent's files

**Return type**
None

**torrents_set_save_path**(*save_path=None*, *torrent_hashes=None*, ***kwargs*) → None

Set the Save Path for one or more torrents.

This method was introduced with qBittorrent v4.4.0 (Web API v2.8.4).

**Raises**

- ***Forbidden403Error*** – cannot write to directory
- ***Conflict409Error*** – cannot create directory

**Parameters**

- **save_path** (Optional[str]) – file path to save torrent contents
- **torrent_hashes** (Optional[Iterable[str]]) – single torrent hash or list of torrent hashes. Or `all` for all torrents.

**Return type**
None

**torrents_set_share_limits**(*ratio_limit=None*, *seeding_time_limit=None*, *inactive_seeding_time_limit=None*, *torrent_hashes=None*, ***kwargs*) → None

Set share limits for one or more torrents.

This method was introduced with qBittorrent v4.1.1 (Web API v2.0.1).

**Parameters**

- **torrent_hashes** (Optional[Iterable[str]]) – single torrent hash or list of torrent hashes. Or `all` for all torrents.
- **ratio_limit** (UnionType[str, int, None]) – max ratio to seed a torrent. (-2 means use the global value and -1 is no limit)
- **seeding_time_limit** (UnionType[str, int, None]) – minutes (-2 means use the global value and -1 is no limit)
- **inactive_seeding_time_limit** (UnionType[str, int, None]) – minutes (-2 means use the global value and -1 is no limit) (added in Web API v2.9.2)

**Return type**
None

**torrents_set_super_seeding**(*enable=None*, *torrent_hashes=None*, ***kwargs*) → None

Set one or more torrents as super seeding.

**Parameters**

- **torrent_hashes** (Optional[Iterable[str]]) – single torrent hash or list of torrent hashes. Or `all` for all torrents.
- **enable** (Optional[bool]) – Defaults to `True` if `None` or unset; `False` to disable

**Return type**
None

**torrents_set_upload_limit**(*limit=None*, *torrent_hashes=None*, ***kwargs*) → None

Set the upload limit for one or more torrents.

**Parameters**

- **torrent_hashes** (Optional[Iterable[str]]) – single torrent hash or list of torrent hashes. Or `all` for all torrents.
- **limit** (UnionType[str, int, None]) – bytes/second (-1 sets the limit to infinity)

**Return type**
None

**torrents_tags**(***kwargs*) → *TagList*

Retrieve all tag definitions.

This method was introduced with qBittorrent v4.2.0 (Web API v2.3.0).

**Return type**
*TagList*

**torrents_toggle_first_last_piece_priority**(*torrent_hashes=None*, ***kwargs*) → None

Toggle priority of first/last piece downloading.

**Parameters**
**torrent_hashes** (Optional[Iterable[str]]) – single torrent hash or list of torrent hashes. Or `all` for all torrents.

**Return type**
None

**torrents_toggle_sequential_download**(*torrent_hashes=None*, ***kwargs*) → None

Toggle sequential download for one or more torrents.

**Parameters**
**torrent_hashes** (Optional[Iterable[str]]) – single torrent hash or list of torrent hashes. Or `all` for all torrents.

**Return type**
None

**torrents_top_priority**(*torrent_hashes=None*, ***kwargs*) → None

Set torrent as highest priority. Torrent Queuing must be enabled.

**Raises**
*Conflict409Error* –

**Parameters**
**torrent_hashes** (Optional[Iterable[str]]) – single torrent hash or list of torrent hashes. Or `all` for all torrents.

**Return type**
None

**torrents_trackers**(*torrent_hash=None*, ***kwargs*) → *TrackersList*

Retrieve individual torrent's trackers. Tracker status is defined in *TrackerStatus*.

**Raises**
*NotFound404Error* –

**Parameters**
**torrent_hash** (Optional[str]) – hash for torrent

**Return type**
*TrackersList*

**torrents_upload_limit**(*torrent_hashes=None*, ***kwargs*) → *TorrentLimitsDictionary*

Retrieve the upload limit for one or more torrents.

**Parameters**

**torrent_hashes** (Optional[Iterable[str]]) – single torrent hash or list of torrent hashes. Or `all` for all torrents.

**Return type**

*TorrentLimitsDictionary*

**torrents_webseeds**(*torrent_hash=None*, ***kwargs*) → *WebSeedsList*

Retrieve individual torrent's web seeds.

**Raises**

***NotFound404Error*** –

**Parameters**

**torrent_hash** (Optional[str]) – hash for torrent

**Return type**

*WebSeedsList*

**class Torrents**(*client*) → None

Allows interaction with the `Torrents` API endpoints.

**Usage**

```
>>> from qbittorrentapi import Client
>>> client = Client(host="localhost:8080", username="admin", password=
↪"adminadmin")
>>> # these are all the same attributes that are available as named in␣
↪the
>>> #  endpoints or the more pythonic names in Client (with or without
↪'torrents_' prepended)
>>> torrent_list = client.torrents.info()
>>> torrent_list_active = client.torrents.info.active()
>>> torrent_list_active_partial = client.torrents.info.active(limit=100,
↪ offset=200)
>>> torrent_list_downloading = client.torrents.info.downloading()
>>> # torrent looping
>>> for torrent in client.torrents.info.completed()
>>> # all torrents endpoints with a 'hashes' parameters support all␣
↪method to apply action to all torrents
>>> client.torrents.pause.all()
>>> client.torrents.resume.all()
>>> # or specify the individual hashes
>>> client.torrents.downloadLimit(torrent_hashes=["...", "..."])
```

**add**(*urls=None*, *torrent_files=None*, *save_path=None*, *cookie=None*, *category=None*, *is_skip_checking=None*, *is_paused=None*, *is_root_folder=None*, *rename=None*, *upload_limit=None*, *download_limit=None*, *use_auto_torrent_management=None*, *is_sequential_download=None*, *is_first_last_piece_priority=None*, *tags=None*, *content_layout=None*, *ratio_limit=None*, *seeding_time_limit=None*, *download_path=None*, *use_download_path=None*, *stop_condition=None*, ***kwargs*) → str

Add one or more torrents by URLs and/or torrent files.

Returns `Ok.` for success and `Fails.` for failure.

**Raises**

- *UnsupportedMediaType415Error* – if file is not a valid torrent file
- *TorrentFileNotFoundError* – if a torrent file doesn't exist
- *TorrentFilePermissionError* – if read permission is denied to torrent file

**Parameters**

- **urls** (Optional[Iterable[str]]) – single instance or an iterable of URLs (`http://`, `https://`, `magnet:`, `bc://bt/`)
- **torrent_files** (Optional[TypeVar(TorrentFilesT, bytes, str, IO[bytes], Mapping[str, Union[bytes, str, IO[bytes]]], Iterable[Union[bytes, str, IO[bytes]]])]) – several options are available to send torrent files to qBittorrent:
  - single instance of bytes: useful if torrent file already read from disk or downloaded from internet.
  - single instance of file handle to torrent file: use `open(<filepath>, 'rb')` to open the torrent file.
  - single instance of a filepath to torrent file: e.g. `/home/user/torrent_filename.torrent`
  - an iterable of the single instances above to send more than one torrent file
  - dictionary with key/value pairs of torrent name and single instance of above object

  Note: The torrent name in a dictionary is useful to identify which torrent file errored. qBittorrent provides back that name in the error text. If a torrent name is not provided, then the name of the file will be used. And in the case of bytes (or if filename cannot be determined), the value 'torrent__n' will be used.
- **save_path** (Optional[str]) – location to save the torrent data
- **cookie** (Optional[str]) – cookie to retrieve torrents by URL
- **category** (Optional[str]) – category to assign to torrent(s)
- **is_skip_checking** (Optional[bool]) – `True` to skip hash checking
- **is_paused** (Optional[bool]) – `True` to add the torrent(s) without starting their downloading
- **is_root_folder** (Optional[bool]) – `True` or `False` to create root folder (superseded by content_layout with v4.3.2)
- **rename** (Optional[str]) – new name for torrent(s)
- **upload_limit** (UnionType[str, int, None]) – upload limit in bytes/second
- **download_limit** (UnionType[str, int, None]) – download limit in bytes/second
- **use_auto_torrent_management** (Optional[bool]) – `True` or `False` to use automatic torrent management
- **is_sequential_download** (Optional[bool]) – `True` or `False` for sequential download
- **is_first_last_piece_priority** (Optional[bool]) – `True` or `False` for first and last piece download priority
- **tags** (Optional[Iterable[str]]) – tag(s) to assign to torrent(s) (added in Web API 2.6.2)

- **content_layout** (Optional[Literal['Original', 'Subfolder', 'NoSubFolder']]) – Original, Subfolder, or NoSubfolder to control filesystem structure for content (added in Web API 2.7)
- **ratio_limit** (UnionType[str, float, None]) – share limit as ratio of upload amt over download amt; e.g. 0.5 or 2.0 (added in Web API 2.8.1)
- **seeding_time_limit** (UnionType[str, int, None]) – number of minutes to seed torrent (added in Web API 2.8.1)
- **download_path** (Optional[str]) – location to download torrent content before moving to save_path (added in Web API 2.8.4)
- **use_download_path** (Optional[bool]) – True or False whether download_path should be used... defaults to True if download_path is specified (added in Web API 2.8.4)
- **stop_condition** (Optional[Literal['MetadataReceived', 'FilesChecked']]) – MetadataReceived or FilesChecked to stop the torrent when started automatically (added in Web API 2.8.15)

**Return type**
str

**add_trackers**(*torrent_hash=None*, *urls=None*, ***kwargs*) → None

Add trackers to a torrent.

**Raises**
*NotFound404Error* –

**Parameters**
- **torrent_hash** (Optional[str]) – hash for torrent
- **urls** (Optional[Iterable[str]]) – tracker URLs to add to torrent

**Return type**
None

**count**() → int

Retrieve count of torrents.

**Return type**
int

**edit_tracker**(*torrent_hash=None*, *original_url=None*, *new_url=None*, ***kwargs*) → None

Replace a torrent's tracker with a different one.

This method was introduced with qBittorrent v4.1.4 (Web API v2.2.0).

**Raises**
- *InvalidRequest400Error* –
- *NotFound404Error* –
- *Conflict409Error* –

**Parameters**
- **torrent_hash** (Optional[str]) – hash for torrent
- **original_url** (Optional[str]) – URL for existing tracker
- **new_url** (Optional[str]) – new URL to replace

**Return type**
None

**export**(*torrent_hash=None*, ***kwargs*) → bytes

Export a .torrent file for the torrent.

This method was introduced with qBittorrent v4.5.0 (Web API v2.8.14).

**Raises**

- ***NotFound404Error*** – torrent not found
- ***Conflict409Error*** – unable to export .torrent file

**Parameters**
**torrent_hash** (Optional[str]) – hash for torrent

**Return type**
bytes

**file_priority**(*torrent_hash=None*, *file_ids=None*, *priority=None*, ***kwargs*) → None

Set priority for one or more files.

**Raises**

- ***InvalidRequest400Error*** – if priority is invalid or at least one file ID is not an integer
- ***NotFound404Error*** –
- ***Conflict409Error*** – if torrent metadata has not finished downloading or at least one file was not found

**Parameters**

- **torrent_hash** (Optional[str]) – hash for torrent
- **file_ids** (Union[int, Iterable[str | int], None]) – single file ID or a list.
- **priority** (UnionType[str, int, None]) – priority for file(s) - https://github.com/qbittorrent/qBittorrent/wiki/WebUI-API-(qBittorrent-4.1)#user-content-set-file-priority

**Return type**
None

**files**(*torrent_hash=None*, ***kwargs*) → *TorrentFilesList*

Retrieve individual torrent's files.

**Raises**
***NotFound404Error*** –

**Parameters**
**torrent_hash** (Optional[str]) – hash for torrent

**Return type**
*TorrentFilesList*

**piece_hashes**(*torrent_hash=None*, ***kwargs*) → *TorrentPieceInfoList*

Retrieve individual torrent's pieces' hashes.

**Raises**
***NotFound404Error*** –

**Parameters**
**torrent_hash** (Optional[str]) – hash for torrent

**Return type**
*TorrentPieceInfoList*

**piece_states**(*torrent_hash=None*, ***kwargs*) → *TorrentPieceInfoList*

Retrieve individual torrent's pieces' states.

**Raises**
***NotFound404Error*** –

**Parameters**
**torrent_hash** (Optional[str]) – hash for torrent

**Return type**
*TorrentPieceInfoList*

**properties**(*torrent_hash=None*, ***kwargs*) → *TorrentPropertiesDictionary*

Retrieve individual torrent's properties.

**Raises**
***NotFound404Error*** –

**Parameters**
**torrent_hash** (Optional[str]) – hash for torrent

**Return type**
*TorrentPropertiesDictionary*

**remove_trackers**(*torrent_hash=None*, *urls=None*, ***kwargs*) → None

Remove trackers from a torrent.

This method was introduced with qBittorrent v4.1.4 (Web API v2.2.0).

**Raises**

- ***NotFound404Error*** –
- ***Conflict409Error*** –

**Parameters**

- **torrent_hash** (Optional[str]) – hash for torrent
- **urls** (Optional[Iterable[str]]) – tracker URLs to removed from torrent

**Return type**
None

**rename**(*torrent_hash=None*, *new_torrent_name=None*, ***kwargs*) → None

Rename a torrent.

**Raises**
***NotFound404Error*** –

**Parameters**

- **torrent_hash** (Optional[str]) – hash for torrent
- **new_torrent_name** (Optional[str]) – new name for torrent

**Return type**
None

**rename_file**(*torrent_hash=None*, *file_id=None*, *new_file_name=None*, *old_path=None*, *new_path=None*, ***kwargs*) → None

Rename a torrent file.

This method was introduced with qBittorrent v4.2.1 (Web API v2.4.0).

**Raises**

- *MissingRequiredParameters400Error* –
- *NotFound404Error* –
- *Conflict409Error* –

**Parameters**

- **torrent_hash** (Optional[str]) – hash for torrent
- **file_id** (UnionType[str, int, None]) – id for file (removed in Web API 2.7)
- **new_file_name** (Optional[str]) – new name for file (removed in Web API 2.7)
- **old_path** (Optional[str]) – path of file to rename (added in Web API 2.7)
- **new_path** (Optional[str]) – new path of file to rename (added in Web API 2.7)

**Return type**
None

**rename_folder**(*torrent_hash=None*, *old_path=None*, *new_path=None*, ***kwargs*) → None

Rename a torrent folder.

This method was introduced with qBittorrent v4.3.2 (Web API v2.7).

**Raises**

- *MissingRequiredParameters400Error* –
- *NotFound404Error* –
- *Conflict409Error* –

**Parameters**

- **torrent_hash** (Optional[str]) – hash for torrent
- **old_path** (Optional[str]) – path of file to rename (added in Web API 2.7)
- **new_path** (Optional[str]) – new path of file to rename (added in Web API 2.7)

**Return type**
None

**trackers**(*torrent_hash=None*, ***kwargs*) → *TrackersList*

Retrieve individual torrent's trackers. Tracker status is defined in *TrackerStatus*.

**Raises**
*NotFound404Error* –

**Parameters**
**torrent_hash** (Optional[str]) – hash for torrent

**Return type**
*TrackersList*

**webseeds**(*torrent_hash=None*, ***kwargs*) → *WebSeedsList*

Retrieve individual torrent's web seeds.

**Raises**
*NotFound404Error* –

**Parameters**
**torrent_hash** (Optional[str]) – hash for torrent

**Return type**
*WebSeedsList*

**class TorrentDictionary**(*data*, *client*) → None

Bases: *ClientCache*[*TorrentsAPIMixIn*], *ListEntry*

Item in *TorrentInfoList*. Allows interaction with individual torrents via the `Torrents` API endpoints.

**Usage**

```
>>> from qbittorrentapi import Client
>>> client = Client(host="localhost:8080", username="admin", password=
↪"adminadmin")
>>> # these are all the same attributes that are available as named in␣
↪the
>>> #  endpoints or the more pythonic names in Client (with or without
↪'transfer_' prepended)
>>> torrent = client.torrents.info()[0]
>>> torrent_hash = torrent.info.hash
>>> # Attributes without inputs and a return value are properties
>>> properties = torrent.properties
>>> trackers = torrent.trackers
>>> files = torrent.files
>>> # Action methods
>>> torrent.edit_tracker(original_url="...", new_url="...")
>>> torrent.remove_trackers(urls="http://127.0.0.2/")
>>> torrent.rename(new_torrent_name="...")
>>> torrent.resume()
>>> torrent.pause()
>>> torrent.recheck()
>>> torrent.torrents_top_priority()
>>> torrent.setLocation(location="/home/user/torrents/")
>>> torrent.setCategory(category="video")
```

**add_tags**(*tags=None*, *torrent_hashes=None*, ***kwargs*) → None

Add one or more tags to one or more torrents.

Note: Tags that do not exist will be created on-the-fly.

This method was introduced with qBittorrent v4.2.0 (Web API v2.3.0).

**Parameters**

- **tags** (Optional[Iterable[str]]) – tag name or list of tags
- **torrent_hashes** (Optional[Iterable[str]]) – single torrent hash or list of torrent hashes. Or `all` for all torrents.

**Return type**
None

**add_trackers**(*torrent_hash=None*, *urls=None*, ***kwargs*) → None

Add trackers to a torrent.

**Raises**
***NotFound404Error*** –

**Parameters**

- **torrent_hash** (Optional[str]) – hash for torrent
- **urls** (Optional[Iterable[str]]) – tracker URLs to add to torrent

**Return type**
None

**bottom_priority**(*torrent_hashes=None*, ***kwargs*) → None

Set torrent as lowest priority. Torrent Queuing must be enabled.

**Raises**
***Conflict409Error*** –

**Parameters**
**torrent_hashes** (Optional[Iterable[str]]) – single torrent hash or list of torrent hashes. Or `all` for all torrents.

**Return type**
None

**decrease_priority**(*torrent_hashes=None*, ***kwargs*) → None

Decrease the priority of a torrent. Torrent Queuing must be enabled.

**Raises**
***Conflict409Error*** –

**Parameters**
**torrent_hashes** (Optional[Iterable[str]]) – single torrent hash or list of torrent hashes. Or `all` for all torrents.

**Return type**
None

**delete**(*delete_files=False*, *torrent_hashes=None*, ***kwargs*) → None

Remove a torrent from qBittorrent and optionally delete its files.

**Parameters**

- **torrent_hashes** (Optional[Iterable[str]]) – single torrent hash or list of torrent hashes. Or `all` for all torrents.
- **delete_files** (Optional[bool]) – True to delete the torrent's files

**Return type**
None

**property download_limit:** ***TorrentLimitsDictionary***

Retrieve the download limit for one or more torrents.

**edit_tracker**(*torrent_hash=None*, *original_url=None*, *new_url=None*, ***kwargs*) → None

Replace a torrent's tracker with a different one.

This method was introduced with qBittorrent v4.1.4 (Web API v2.2.0).

**Raises**

- *InvalidRequest400Error* –
- *NotFound404Error* –
- *Conflict409Error* –

**Parameters**

- **torrent_hash** (Optional[str]) – hash for torrent
- **original_url** (Optional[str]) – URL for existing tracker
- **new_url** (Optional[str]) – new URL to replace

**Return type**
None

**export**(*torrent_hash=None*, ***kwargs*) → bytes

Export a .torrent file for the torrent.

This method was introduced with qBittorrent v4.5.0 (Web API v2.8.14).

**Raises**

- *NotFound404Error* – torrent not found
- *Conflict409Error* – unable to export .torrent file

**Parameters**
**torrent_hash** (Optional[str]) – hash for torrent

**Return type**
bytes

**file_priority**(*torrent_hash=None*, *file_ids=None*, *priority=None*, ***kwargs*) → None

Set priority for one or more files.

**Raises**

- *InvalidRequest400Error* – if priority is invalid or at least one file ID is not an integer
- *NotFound404Error* –
- *Conflict409Error* – if torrent metadata has not finished downloading or at least one file was not found

**Parameters**

- **torrent_hash** (Optional[str]) – hash for torrent
- **file_ids** (Union[int, Iterable[str | int], None]) – single file ID or a list.
- **priority** (UnionType[str, int, None]) – priority for file(s) - https://github.com/qbittorrent/qBittorrent/wiki/WebUI-API-(qBittorrent-4.1)#user-content-set-file-priority

**Return type**
None

**property files:** *TorrentFilesList*

Retrieve individual torrent's files.

**Raises**
*NotFound404Error* –

**Parameters**
**torrent_hash** (Optional[str]) – hash for torrent

**increase_priority**(*torrent_hashes=None*, ***kwargs*) → None

Increase the priority of a torrent. Torrent Queuing must be enabled.

**Raises**
*Conflict409Error* –

**Parameters**
**torrent_hashes** (Optional[Iterable[str]]) – single torrent hash or list of torrent hashes. Or `all` for all torrents.

**Return type**
None

**property info:** *TorrentDictionary*

Returns data from *torrents_info()* for the torrent.

**pause**(*torrent_hashes=None*, ***kwargs*) → None

Pause one or more torrents in qBittorrent.

**Parameters**
**torrent_hashes** (Optional[Iterable[str]]) – single torrent hash or list of torrent hashes. Or `all` for all torrents.

**Return type**
None

**property piece_hashes:** *TorrentPieceInfoList*

Retrieve individual torrent's pieces' hashes.

**Raises**
*NotFound404Error* –

**Parameters**
**torrent_hash** (Optional[str]) – hash for torrent

**property piece_states:** *TorrentPieceInfoList*

Retrieve individual torrent's pieces' states.

**Raises**
*NotFound404Error* –

**Parameters**
**torrent_hash** (Optional[str]) – hash for torrent

**property properties:** *TorrentPropertiesDictionary*

Retrieve individual torrent's properties.

**Raises**
*NotFound404Error* –

**Parameters**
**torrent_hash** (Optional[str]) – hash for torrent

**reannounce**(*torrent_hashes=None*, ***kwargs*) → None

Reannounce a torrent.

This method was introduced with qBittorrent v4.1.2 (Web API v2.0.2).

**Parameters**
**torrent_hashes** (Optional[Iterable[str]]) – single torrent hash or list of torrent hashes. Or `all` for all torrents.

**Return type**
None

**recheck**(*torrent_hashes=None*, ***kwargs*) → None

Recheck a torrent in qBittorrent.

**Parameters**
**torrent_hashes** (Optional[Iterable[str]]) – single torrent hash or list of torrent hashes. Or `all` for all torrents.

**Return type**
None

**remove_tags**(*tags=None*, *torrent_hashes=None*, ***kwargs*) → None

Add one or more tags to one or more torrents.

This method was introduced with qBittorrent v4.2.0 (Web API v2.3.0).

**Parameters**

- **tags** (Optional[Iterable[str]]) – tag name or list of tags
- **torrent_hashes** (Optional[Iterable[str]]) – single torrent hash or list of torrent hashes. Or `all` for all torrents.

**Return type**
None

**remove_trackers**(*torrent_hash=None*, *urls=None*, ***kwargs*) → None

Remove trackers from a torrent.

This method was introduced with qBittorrent v4.1.4 (Web API v2.2.0).

**Raises**

- *NotFound404Error* –
- *Conflict409Error* –

**Parameters**

- **torrent_hash** (Optional[str]) – hash for torrent
- **urls** (Optional[Iterable[str]]) – tracker URLs to removed from torrent

**Return type**
None

**rename**(*torrent_hash=None*, *new_torrent_name=None*, ***kwargs*) → None

Rename a torrent.

**Raises**
*NotFound404Error* –

**Parameters**

- **torrent_hash** (Optional[str]) – hash for torrent
- **new_torrent_name** (Optional[str]) – new name for torrent

**Return type**
None

**rename_file**(*torrent_hash=None*, *file_id=None*, *new_file_name=None*, *old_path=None*, *new_path=None*, ***kwargs*) → None

Rename a torrent file.

This method was introduced with qBittorrent v4.2.1 (Web API v2.4.0).

**Raises**

- ***MissingRequiredParameters400Error*** –
- ***NotFound404Error*** –
- ***Conflict409Error*** –

**Parameters**

- **torrent_hash** (Optional[str]) – hash for torrent
- **file_id** (UnionType[str, int, None]) – id for file (removed in Web API 2.7)
- **new_file_name** (Optional[str]) – new name for file (removed in Web API 2.7)
- **old_path** (Optional[str]) – path of file to rename (added in Web API 2.7)
- **new_path** (Optional[str]) – new path of file to rename (added in Web API 2.7)

**Return type**
None

**rename_folder**(*torrent_hash=None*, *old_path=None*, *new_path=None*, ***kwargs*) → None

Rename a torrent folder.

This method was introduced with qBittorrent v4.3.2 (Web API v2.7).

**Raises**

- ***MissingRequiredParameters400Error*** –
- ***NotFound404Error*** –
- ***Conflict409Error*** –

**Parameters**

- **torrent_hash** (Optional[str]) – hash for torrent
- **old_path** (Optional[str]) – path of file to rename (added in Web API 2.7)
- **new_path** (Optional[str]) – new path of file to rename (added in Web API 2.7)

**Return type**
None

**resume**(*torrent_hashes=None*, ***kwargs*) → None

Resume one or more torrents in qBittorrent.

**Parameters**
**torrent_hashes** (Optional[Iterable[str]]) – single torrent hash or list of torrent hashes. Or `all` for all torrents.

**Return type**
None

**setDownloadPath**(*download_path=None*, *torrent_hashes=None*, ***kwargs*) → None

Set the Download Path for one or more torrents.

This method was introduced with qBittorrent v4.4.0 (Web API v2.8.4).

**Raises**

- ***Forbidden403Error*** – cannot write to directory
- ***Conflict409Error*** – cannot create directory

**Parameters**

- **download_path** (Optional[str]) – file path to save torrent contents before torrent finishes downloading
- **torrent_hashes** (Optional[Iterable[str]]) – single torrent hash or list of torrent hashes. Or `all` for all torrents.

**Return type**
None

**setSavePath**(*save_path=None*, *torrent_hashes=None*, ***kwargs*) → None

Set the Save Path for one or more torrents.

This method was introduced with qBittorrent v4.4.0 (Web API v2.8.4).

**Raises**

- ***Forbidden403Error*** – cannot write to directory
- ***Conflict409Error*** – cannot create directory

**Parameters**

- **save_path** (Optional[str]) – file path to save torrent contents
- **torrent_hashes** (Optional[Iterable[str]]) – single torrent hash or list of torrent hashes. Or `all` for all torrents.

**Return type**
None

**set_auto_management**(*enable=None*, *torrent_hashes=None*, ***kwargs*) → None

Enable or disable automatic torrent management for one or more torrents.

**Parameters**

- **torrent_hashes** (Optional[Iterable[str]]) – single torrent hash or list of torrent hashes. Or `all` for all torrents.
- **enable** (Optional[bool]) – Defaults to `True` if `None` or unset; use `False` to disable

**Return type**
None

**set_category**(*category=None*, *torrent_hashes=None*, ***kwargs*) → None

Set a category for one or more torrents.

**Raises**
***Conflict409Error*** – for bad category

**Parameters**

- **torrent_hashes** (Optional[Iterable[str]]) – single torrent hash or list of torrent hashes. Or `all` for all torrents.
- **category** (Optional[str]) – category to assign to torrent

**Return type**
None

**set_download_limit**(*limit=None*, *torrent_hashes=None*, ***kwargs*) → None

Set the download limit for one or more torrents.

**Parameters**

- **torrent_hashes** (Optional[Iterable[str]]) – single torrent hash or list of torrent hashes. Or `all` for all torrents.
- **limit** (UnionType[str, int, None]) – bytes/second (-1 sets the limit to infinity)

**Return type**
None

**set_download_path**(*download_path=None*, *torrent_hashes=None*, ***kwargs*) → None

Set the Download Path for one or more torrents.

This method was introduced with qBittorrent v4.4.0 (Web API v2.8.4).

**Raises**

- ***Forbidden403Error*** – cannot write to directory
- ***Conflict409Error*** – cannot create directory

**Parameters**

- **download_path** (Optional[str]) – file path to save torrent contents before torrent finishes downloading
- **torrent_hashes** (Optional[Iterable[str]]) – single torrent hash or list of torrent hashes. Or `all` for all torrents.

**Return type**
None

**set_force_start**(*enable=None*, *torrent_hashes=None*, ***kwargs*) → None

Force start one or more torrents.

**Parameters**

- **torrent_hashes** (Optional[Iterable[str]]) – single torrent hash or list of torrent hashes. Or `all` for all torrents.
- **enable** (Optional[bool]) – Defaults to `True` if `None` or unset; `False` is equivalent to *torrents_resume()*.

**Return type**
None

**set_location**(*location=None*, *torrent_hashes=None*, ***kwargs*) → None

Set location for torrents' files.

**Raises**

- ***Forbidden403Error*** – if the user doesn't have permissions to write to the location (only before v4.5.2 - write check was removed.)
- ***Conflict409Error*** – if the directory cannot be created at the location

**Parameters**

- **torrent_hashes** (Optional[Iterable[str]]) – single torrent hash or list of torrent hashes. Or `all` for all torrents.
- **location** (Optional[str]) – disk location to move torrent's files

**Return type**
None

**set_save_path**(*save_path=None*, *torrent_hashes=None*, ***kwargs*) → None

Set the Save Path for one or more torrents.

This method was introduced with qBittorrent v4.4.0 (Web API v2.8.4).

**Raises**

- ***Forbidden403Error*** – cannot write to directory
- ***Conflict409Error*** – cannot create directory

**Parameters**

- **save_path** (Optional[str]) – file path to save torrent contents
- **torrent_hashes** (Optional[Iterable[str]]) – single torrent hash or list of torrent hashes. Or `all` for all torrents.

**Return type**
None

**set_share_limits**(*ratio_limit=None*, *seeding_time_limit=None*, *inactive_seeding_time_limit=None*, *torrent_hashes=None*, ***kwargs*) → None

Set share limits for one or more torrents.

This method was introduced with qBittorrent v4.1.1 (Web API v2.0.1).

**Parameters**

- **torrent_hashes** (Optional[Iterable[str]]) – single torrent hash or list of torrent hashes. Or `all` for all torrents.
- **ratio_limit** (UnionType[str, int, None]) – max ratio to seed a torrent. (-2 means use the global value and -1 is no limit)
- **seeding_time_limit** (UnionType[str, int, None]) – minutes (-2 means use the global value and -1 is no limit)
- **inactive_seeding_time_limit** (UnionType[str, int, None]) – minutes (-2 means use the global value and -1 is no limit) (added in Web API v2.9.2)

**Return type**
None

**set_super_seeding**(*enable=None*, *torrent_hashes=None*, ***kwargs*) → None

Set one or more torrents as super seeding.

**Parameters**

- **torrent_hashes** (Optional[Iterable[str]]) – single torrent hash or list of torrent hashes. Or `all` for all torrents.
- **enable** (Optional[bool]) – Defaults to `True` if `None` or unset; `False` to disable

**Return type**
None

**set_upload_limit**(*limit=None*, *torrent_hashes=None*, ***kwargs*) → None

Set the upload limit for one or more torrents.

**Parameters**

- **torrent_hashes** (Optional[Iterable[str]]) – single torrent hash or list of torrent hashes. Or `all` for all torrents.
- **limit** (UnionType[str, int, None]) – bytes/second (-1 sets the limit to infinity)

**Return type**
None

**property state_enum: *TorrentState***

Torrent state enum.

**sync_local**() → None

Update local cache of torrent info.

**Return type**
None

**toggle_first_last_piece_priority**(*torrent_hashes=None*, ***kwargs*) → None

Toggle priority of first/last piece downloading.

**Parameters**
**torrent_hashes** (Optional[Iterable[str]]) – single torrent hash or list of torrent hashes. Or `all` for all torrents.

**Return type**
None

**toggle_sequential_download**(*torrent_hashes=None*, ***kwargs*) → None

Toggle sequential download for one or more torrents.

**Parameters**
**torrent_hashes** (Optional[Iterable[str]]) – single torrent hash or list of torrent hashes. Or `all` for all torrents.

**Return type**
None

**top_priority**(*torrent_hashes=None*, ***kwargs*) → None

Set torrent as highest priority. Torrent Queuing must be enabled.

**Raises**
***Conflict409Error*** –

**Parameters**
**torrent_hashes** (Optional[Iterable[str]]) – single torrent hash or list of torrent hashes. Or `all` for all torrents.

**Return type**
None

**property trackers: *TrackersList***

Retrieve individual torrent's trackers. Tracker status is defined in *TrackerStatus*.

**Raises**
***NotFound404Error*** –

**Parameters**
**torrent_hash** (Optional[str]) – hash for torrent

**property upload_limit: *TorrentLimitsDictionary***

Retrieve the upload limit for one or more torrents.

**Parameters**
**torrent_hashes** (Optional[Iterable[str]]) – single torrent hash or list of torrent hashes. Or `all` for all torrents.

**property webseeds:** *WebSeedsList*

Retrieve individual torrent's web seeds.

**Raises**
*NotFound404Error* –

**Parameters**
**torrent_hash** (Optional[str]) – hash for torrent

**class TorrentCategories**(**args*, *client*, ***kwargs*)

Bases: *ClientCache*[*TorrentsAPIMixIn*]

Allows interaction with torrent categories within the `Torrents` API endpoints.

**Usage**

```
>>> from qbittorrentapi import Client
>>> client = Client(host="localhost:8080", username="admin", password=
↪"adminadmin")
>>> # these are all the same attributes that are available as named in␣
↪the
>>> #  endpoints or the more pythonic names in Client (with or without
↪'torrents_' prepended)
>>> categories = client.torrent_categories.categories
>>> # create or edit categories
>>> client.torrent_categories.create_category(name="Video", save_path="/
↪home/user/torrents/Video")
>>> client.torrent_categories.edit_category(name="Video", save_path="/
↪data/torrents/Video")
>>> # edit or create new by assignment
>>> client.torrent_categories.categories = dict(name="Video", save_path=
↪"/hone/user/")
>>> # delete categories
>>> client.torrent_categories.removeCategories(categories="Video")
>>> client.torrent_categories.removeCategories(categories=["Audio",
↪"ISOs"])
```

**property categories:** *TorrentCategoriesDictionary*

Retrieve all category definitions.

This method was introduced with qBittorrent v4.1.4 (Web API v2.1.1).

Note: `torrents/categories` is not available until v2.1.0

**create_category**(*name=None*, *save_path=None*, *download_path=None*, *enable_download_path=None*, ***kwargs*) → None

Create a new torrent category.

**Raises**
*Conflict409Error* – if category name is not valid or unable to create

**Parameters**
- **name** (Optional[str]) – name for new category

- **save_path** (Optional[str]) – location to save torrents for this category (added in Web API 2.1.0)
- **download_path** (Optional[str]) – download location for torrents with this category
- **enable_download_path** (Optional[bool]) – True or False to enable or disable download path

**Return type**
None

**edit_category**(*name=None*, *save_path=None*, *download_path=None*, *enable_download_path=None*, ***kwargs*) → None

Edit an existing category.

This method was introduced with qBittorrent v4.1.3 (Web API v2.1.0).

**Raises**
*Conflict409Error* – if category name is not valid or unable to create

**Parameters**

- **name** (Optional[str]) – category to edit
- **save_path** (Optional[str]) – new location to save files for this category
- **download_path** (Optional[str]) – download location for torrents with this category
- **enable_download_path** (Optional[bool]) – True or False to enable or disable download path

**Return type**
None

**remove_categories**(*categories=None*, ***kwargs*) → None

Delete one or more categories.

**Parameters**
**categories** (Optional[Iterable[str]]) – categories to delete

**Return type**
None

**class TorrentTags**(**args*, *client*, ***kwargs*)

Bases: *ClientCache*[*TorrentsAPIMixIn*]

Allows interaction with torrent tags within the "Torrent" API endpoints.

**Usage:**

```
>>> from qbittorrentapi import Client
>>> client = Client(host="localhost:8080", username="admin", password=
↪"adminadmin")
>>> tags = client.torrent_tags.tags
>>> client.torrent_tags.tags = "tv show"  # create category
>>> client.torrent_tags.create_tags(tags=["tv show", "linux distro"])
>>> client.torrent_tags.delete_tags(tags="tv show")
```

**add_tags**(*tags=None*, *torrent_hashes=None*, ***kwargs*) → None

Add one or more tags to one or more torrents.

Note: Tags that do not exist will be created on-the-fly.

This method was introduced with qBittorrent v4.2.0 (Web API v2.3.0).

**Parameters**

- **tags** (Optional[Iterable[str]]) – tag name or list of tags
- **torrent_hashes** (Optional[Iterable[str]]) – single torrent hash or list of torrent hashes. Or `all` for all torrents.

**Return type**
None

**create_tags**(*tags=None*, ***kwargs*) → None

Create one or more tags.

This method was introduced with qBittorrent v4.2.0 (Web API v2.3.0).

**Parameters**
**tags** (Optional[Iterable[str]]) – tag name or list of tags

**Return type**
None

**delete_tags**(*tags=None*, ***kwargs*) → None

Delete one or more tags.

This method was introduced with qBittorrent v4.2.0 (Web API v2.3.0).

**Parameters**
**tags** (Optional[Iterable[str]]) – tag name or list of tags

**Return type**
None

**remove_tags**(*tags=None*, *torrent_hashes=None*, ***kwargs*) → None

Add one or more tags to one or more torrents.

This method was introduced with qBittorrent v4.2.0 (Web API v2.3.0).

**Parameters**

- **tags** (Optional[Iterable[str]]) – tag name or list of tags
- **torrent_hashes** (Optional[Iterable[str]]) – single torrent hash or list of torrent hashes. Or `all` for all torrents.

**Return type**
None

**property tags:** *TagList*

Retrieve all tag definitions.

This method was introduced with qBittorrent v4.2.0 (Web API v2.3.0).

**class TorrentPropertiesDictionary**(*data=None*, ***kwargs*)

Bases: *Dictionary*[Union[None, int, str, bool, Sequence[JsonValueT], Mapping[str, JsonValueT]]]

Response to *torrents_properties()*

Definition: https://github.com/qbittorrent/qBittorrent/wiki/WebUI-API-(qBittorrent-4.1)#user-content-get-torrent-generic-properties

**class TorrentLimitsDictionary**(*data=None*, ***kwargs*)

Bases: *Dictionary*[Union[None, int, str, bool, Sequence[JsonValueT], Mapping[str, JsonValueT]]]

Response to *torrents_download_limit()*

**class TorrentCategoriesDictionary**(*data=None*, ***kwargs*)

Bases: *Dictionary*[Union[None, int, str, bool, Sequence[JsonValueT], Mapping[str, JsonValueT]]]

Response to *torrents_categories()*

**class TorrentsAddPeersDictionary**(*data=None*, ***kwargs*)

Bases: *Dictionary*[Union[None, int, str, bool, Sequence[JsonValueT], Mapping[str, JsonValueT]]]

Response to *torrents_add_peers()*

**class TorrentFilesList**(*list_entries*, *client=None*)

Bases: *List*[*TorrentFile*]

Response to *torrents_files()*

Definition: https://github.com/qbittorrent/qBittorrent/wiki/WebUI-API-(qBittorrent-4.1)#user-content-get-torrent-contents

**class TorrentFile**(*data=None*, ***kwargs*)

Bases: *ListEntry*

Item in *TorrentFilesList*

**class WebSeedsList**(*list_entries*, *client=None*)

Bases: *List*[*WebSeed*]

Response to *torrents_webseeds()*

Definition: https://github.com/qbittorrent/qBittorrent/wiki/WebUI-API-(qBittorrent-4.1)#user-content-get-torrent-web-seeds

**class WebSeed**(*data=None*, ***kwargs*)

Bases: *ListEntry*

Item in *WebSeedsList*

**class TrackersList**(*list_entries*, *client=None*)

Bases: *List*[*Tracker*]

Response to *torrents_trackers()*

Definition: https://github.com/qbittorrent/qBittorrent/wiki/WebUI-API-(qBittorrent-4.1)#user-content-get-torrent-trackers

**class Tracker**(*data=None*, ***kwargs*)

Bases: *ListEntry*

Item in *TrackersList*

**class TorrentInfoList**(*list_entries*, *client=None*)

Bases: *List*[TorrentDictionary]

Response to *torrents_info()*

Definition: https://github.com/qbittorrent/qBittorrent/wiki/WebUI-API-(qBittorrent-4.1)#user-content-get-torrent-list

**class TorrentPieceInfoList**(*list_entries*, *client=None*)

Bases: *List*[*TorrentPieceData*]

Response to *torrents_piece_states()* and *torrents_piece_hashes()*

**class TorrentPieceData**(*data=None*, ***kwargs*)

Bases: *ListEntry*

Item in *TorrentPieceInfoList*

**class TagList**(*list_entries*, *client=None*)

Bases: *List*[*Tag*]

Response to *torrents_tags()*

**class Tag**(*data=None*, ***kwargs*)

Bases: *ListEntry*

Item in *TagList*

### Transfer

**class TransferAPIMixIn**(*host=None*, *port=None*, *username=None*, *password=None*, *EXTRA_HEADERS=None*, *REQUESTS_ARGS=None*, *VERIFY_WEBUI_CERTIFICATE=True*, *FORCE_SCHEME_FROM_HOST=False*, *RAISE_NOTIMPLEMENTEDERROR_FOR_UNIMPLEMENTED_API_ENDPOINTS=False*, *RAISE_ERROR_FOR_UNSUPPORTED_QBITTORRENT_VERSIONS=False*, *VERBOSE_RESPONSE_LOGGING=False*, *SIMPLE_RESPONSES=False*, *DISABLE_LOGGING_DEBUG_OUTPUT=False*) → None

Bases: *AppAPIMixIn*

Implementation of all `Transfer` API methods.

**Usage**

```
>>> from qbittorrentapi import Client
>>> client = Client(host="localhost:8080", username="admin", password=
↪"adminadmin")
>>> transfer_info = client.transfer_info()
>>> client.transfer_set_download_limit(limit=1024000)
```

**transfer_ban_peers**(*peers=None*, ***kwargs*) → None

Ban one or more peers.

This method was introduced with qBittorrent v4.2.0 (Web API v2.3.0).

**Parameters**

**peers** (Union[str, Iterable[str], None]) – one or more peers to ban. each peer should take the form 'host:port'

**Return type**

None

**transfer_download_limit**(***kwargs*) → int

Retrieves download limit; 0 is unlimited.

**Return type**

int

**transfer_info**(***kwargs*) → *TransferInfoDictionary*

Retrieves the global transfer info found in qBittorrent status bar.

**Return type**
*TransferInfoDictionary*

**transfer_setSpeedLimitsMode**(*intended_state=None*, ***kwargs*) → None

Sets whether alternative speed limits are enabled.

**Parameters**
**intended_state** (Optional[bool]) – True to enable alt speed and False to disable. Leaving None will toggle the current state.

**Return type**
None

**transfer_set_download_limit**(*limit=None*, ***kwargs*) → None

Set the global download limit in bytes/second.

**Parameters**
**limit** (UnionType[str, int, None]) – download limit in bytes/second (0 or -1 for no limit)

**Return type**
None

**transfer_set_speed_limits_mode**(*intended_state=None*, ***kwargs*) → None

Sets whether alternative speed limits are enabled.

**Parameters**
**intended_state** (Optional[bool]) – True to enable alt speed and False to disable. Leaving None will toggle the current state.

**Return type**
None

**transfer_set_upload_limit**(*limit=None*, ***kwargs*) → None

Set the global download limit in bytes/second.

**Parameters**
**limit** (UnionType[str, int, None]) – upload limit in bytes/second (0 or -1 for no limit)

**Return type**
None

**transfer_speed_limits_mode**(***kwargs*) → str

Returns 1 if alternative speed limits are currently enabled, **0** otherwise.

**Return type**
str

**transfer_toggle_speed_limits_mode**(*intended_state=None*, ***kwargs*) → None

Sets whether alternative speed limits are enabled.

**Parameters**
**intended_state** (Optional[bool]) – True to enable alt speed and False to disable. Leaving None will toggle the current state.

**Return type**
None

**transfer_upload_limit**(***kwargs*) → int

Retrieves upload limit; 0 is unlimited.

**Return type**
int

**class Transfer**(**args*, *client*, ***kwargs*)

Allows interaction with the `Transfer` API endpoints.

**Usage**

```
>>> from qbittorrentapi import Client
>>> client = Client(host="localhost:8080", username="admin", password=
↪"adminadmin")
>>> # these are all the same attributes that are available as named in␣
↪the
>>> #  endpoints or the more pythonic names in Client (with or without
↪'transfer_' prepended)
>>> transfer_info = client.transfer.info
>>> # access and set download/upload limits as attributes
>>> dl_limit = client.transfer.download_limit
>>> # this updates qBittorrent in real-time
>>> client.transfer.download_limit = 1024000
>>> # update speed limits mode to alternate or not
>>> client.transfer.speedLimitsMode = True
```

**ban_peers**(*peers=None*, ***kwargs*) → None

Ban one or more peers.

This method was introduced with qBittorrent v4.2.0 (Web API v2.3.0).

**Parameters**
**peers** (Union[str, Iterable[str], None]) – one or more peers to ban. each peer should take the form 'host:port'

**Return type**
None

**property download_limit: int**

Retrieves download limit; 0 is unlimited.

**property info: *TransferInfoDictionary***

Retrieves the global transfer info found in qBittorrent status bar.

**set_download_limit**(*limit=None*, ***kwargs*) → None

Set the global download limit in bytes/second.

**Parameters**
**limit** (UnionType[str, int, None]) – download limit in bytes/second (0 or -1 for no limit)

**Return type**
None

**set_speed_limits_mode**(*intended_state=None*, ***kwargs*) → None

Sets whether alternative speed limits are enabled.

**Parameters**
**intended_state** (Optional[bool]) – True to enable alt speed and False to disable. Leaving None will toggle the current state.

**Return type**
None

`set_upload_limit`(*limit=None*, ***kwargs*) → None

Set the global download limit in bytes/second.

**Parameters**
**limit** (UnionType[str, int, None]) – upload limit in bytes/second (0 or -1 for no limit)

**Return type**
None

`property speed_limits_mode: str`

Returns 1 if alternative speed limits are currently enabled, `0` otherwise.

`toggle_speed_limits_mode`(*intended_state=None*, ***kwargs*) → None

Sets whether alternative speed limits are enabled.

**Parameters**
**intended_state** (Optional[bool]) – True to enable alt speed and False to disable. Leaving None will toggle the current state.

**Return type**
None

`property upload_limit: int`

Retrieves upload limit; 0 is unlimited.

`class TransferInfoDictionary`(*data=None*, ***kwargs*)

Bases: *Dictionary*[Union[None, int, str, bool, Sequence[JsonValueT], Mapping[str, JsonValueT]]]

Response to *transfer_info()*

Definition: https://github.com/qbittorrent/qBittorrent/wiki/WebUI-API-(qBittorrent-4.1)#user-content-get-global-transfer-info

## Version

`class Version`

Allows introspection for whether this Client supports different versions of the qBittorrent application and its Web API.

Note that if a version is not listed as "supported" here, many (if not all) methods are likely to function properly since the Web API is largely backwards and forward compatible…albeit with some notable exceptions.

`classmethod is_api_version_supported`(*api_version*) → bool

Returns whether a version of the qBittorrent Web API is fully supported by this API client.

**Parameters**
**api_version** (str) – version of qBittorrent Web API version such as `2.8.4`

**Return type**
bool

**Returns**
`True` or `False` for whether version is supported

**classmethod is_app_version_supported**(*app_version*) → bool

Returns whether a version of the qBittorrent application is fully supported by this API client.

**Parameters**
**app_version** (str) – version of qBittorrent application such as v4.4.0

**Return type**
bool

**Returns**
`True` or `False` for whether version is supported

**classmethod latest_supported_api_version**() → str

Returns the most recent version of qBittorrent Web API that is supported.

**Return type**
str

**classmethod latest_supported_app_version**() → str

Returns the most recent version of qBittorrent that is supported.

**Return type**
str

**classmethod supported_api_versions**() → set[str]

Set of all supported qBittorrent Web API versions.

**Return type**
set[str]

**classmethod supported_app_versions**() → set[str]

Set of all supported qBittorrent application versions.

**Return type**
set[str]

# PYTHON MODULE INDEX

## q

# INDEX

## Symbols



## A



## B



## C



## D

## E



## F



## H



## I



## J



## L



## M

## N



## P



## Q



## R



## S

## T

## U



## V



## W