Get Started Products Communit

Web API

Community Developers Web API

Table of Contents

What is the WebUI?

It is functionality built into µTorrent for almost complete control and configuration of µTorrent from both local and remote applications, as well as display of most data available. Normally, this functionality is used in a web browser with the WebUI we provide, but any application can talk to µTorrent directly by using the Web API.

What is the Web API?

It is an API to access the functions of the WebUI built into µTorrent. This is independent of the standard WebUI served up by µTorrent and requires nothing on the client-side besides the option being turned on.

The API is stable and largely complete. Missing functionality will be added over time and compatibility with existing applications will generally be preserved, so little to no work will be needed to keep your application working with future versions of µTorrent.

The current WebUI as well as select projects making use of the API are available on Github. Important updates to the API will be mentioned both here and there.

Many other projects are available in the WebUI section of the forums. You can post your project there as well!

General notes for API access

Once enabled in the µTorrent settings (Preferences -> WebUI), the base URI to access it is http://[IP]:[PORT]/gui/. The data returned by calls is in the JSON format.

Authentication is done with basic HTTP authentication. The guest account, if enabled, is limited to a subset of the calls. Action calls that modify torrent state and getsettings are disallowed.

Unless otherwise noted, each call is done using HTTP GET. Parameters are added onto the base URI like normal. The first parameter should be always the command (e.g. ?list or ?action).

Most action commands require a hash to be passed. This is the infohash of the torrent, obtained from listing all torrents. Each hash is a 40-character ASCII string. Some commands accept multiple infohashes chained together, e.g. http://[IP]:[PORT]/gui/?action=[ACTION]&hash=[TORRENT HASH 1]&hash=[TORRENT HASH 2]&... to cut down on the number of requests required.

When setting boolean values either by ?action=setsetting or ?action=setprops, the value parameter should be sent as 0 for "false" or 1 for "true" rather than a string indicating "true" or "false".

Token Authentication System (important!)

A token authentication system was implemented in µTorrent to prevent cross-site request forgeries

A token authentication system was implemented in µTorrent to prevent cross-site request forgeries (CSRF). **All developers creating applications that use the WebUI backend MUST implement support for this system**, as the application will otherwise fail if the user has webui.token_auth enabled.

In 1.8.3 and earlier, token authentication is disabled by default (though users can enable it manually), but this option WILL be enabled by default in future versions, so implementing support now is a requirement for future compatibility.

Modifying Settings

The base parameter for settings is ?action=getsettings. Using this parameter by itself will give a list of settings in the following format:

```
{
         "build": BUILD NUMBER (integer),
         "settings": [
         [
         OPTION NAME (string),
         TYPE* (integer),
         VALUE (string)  ],
         ...
         ]
         }
```

OPTION NAME is the name of the setting. They are not listed here, as some of the settings (particularly advanced ones) vary with each version and most are self-explanatory. However, a near-complete list for 1.8.2 is given here for your perusal.

* TYPE: The TYPE is an integer value that indicates what type of data is enclosed within the VALUE string. The following is a list of the possible TYPEs and what VALUE type it corresponds to:

- 0 = Integer
- 1 = Boolean
- 2 = String

To change settings, the following URI can be used: http://[IP]:[PORT]/gui/?action=setsetting&s=[SETTING]&v=[VALUE]

Multiple settings can be changed in a single request by chaining together s= and v= in the URI. s= defines the setting and v= is the value for the s= immediately preceding it.

For example, to set the global upload cap to 10KiB/s and the global download cap to 40KiB/s, you

For example, to set the global upload cap to 10KiB/s and the global download cap to 40KiB/s, you would request this URI: http://[IP]:[PORT]/gui/?action=setsetting&s=max_ul_rate&v=10&s=max_dl_rate&v=40

Torrent/labels List

To get the list of all torrents, request http://[IP]:[PORT]/gui/?list=1 . This will return the torrents in the following fashion:

```
{
        "build": BUILD NUMBER (integer),
        "label": [
        [
        LABEL (string),
        TORRENTS IN LABEL (integer)     ],
        ...
        ],
        "torrents": [
        [
        HASH (string),
        STATUS* (integer),
        NAME (string),
        SIZE (integer in bytes),
        PERCENT PROGRESS (integer in per mils),
        DOWNLOADED (integer in bytes),
        UPLOADED (integer in bytes),
        RATIO (integer in per mils),
        UPLOAD SPEED (integer in bytes per second),
        DOWNLOAD SPEED (integer in bytes per second),
        ETA (integer in seconds),
        LABEL (string),
        PEERS CONNECTED (integer),
        PEERS IN SWARM (integer),
        SEEDS CONNECTED (integer),
        SEEDS IN SWARM (integer),
        AVAILABILITY (integer in 1/65535ths),
        TORRENT QUEUE ORDER (integer),
        REMAINING (integer in bytes)    ],
        ...
        ],
        "torrentc": CACHE ID** (string integer) }
```

* STATUS: The STATUS is a bitfield represented as integers, which is obtained by adding up the different values for corresponding statuses:

- 1 = Started
- 2 = Checking
- 4 = Start after check
- 8 = Checked
- 16 = Error

- 16 = Error
- 32 = Paused
- 64 = Queued
- 128 = Loaded

For example, if a torrent job has a status of 201 = 128 + 64 + 8 + 1, then it is loaded, queued, checked, and started. A bitwise AND operator should be used to determine whether the given STATUS contains a particular status.

** CACHE ID: The CACHE ID is a number randomly generated by µTorrent for the given data. By requesting the torrent list using http://[IP]:[PORT]/gui/?list=1&cid=[CACHE ID], only the items that have changed since the list corresponding to the CACHE ID was sent will be returned. This is used to minimize bandwidth usage and simplify parsing by decreasing the amount of data sent by µTorrent. In this situation, two new dictionary keys replace "torrents," and the returned JSON will look as follows:

```
{
        "build": BUILD NUMBER (integer),
        "label": [
        [
        LABEL (string),
        TORRENTS IN LABEL (integer)     ],
        ...
        ],
        "torrentp": [
        [
        HASH (string),
        STATUS (integer),
        NAME (string),
        SIZE (integer in bytes),
        PERCENT PROGRESS (integer in per mils),
        DOWNLOADED (integer in bytes),
        UPLOADED (integer in bytes),
        RATIO (integer in per mils),
        UPLOAD SPEED (integer in bytes per second),
        DOWNLOAD SPEED (integer in bytes per second),
        ETA (integer in seconds),
        LABEL (string),
        PEERS CONNECTED (integer),
        PEERS IN SWARM (integer),
        SEEDS CONNECTED (integer),
        SEEDS IN SWARM (integer),
        AVAILABILITY (integer in 1/65535ths),
        TORRENT QUEUE ORDER (integer),
        REMAINING (integer in bytes)        ],
        ...
        ],
        "torrentm": [
        HASH (string),
        ...
        ]
        "torrentc": CACHE ID (string integer)   }
```

The "torrentp" array contains a list of torrent jobs that have changed since the corresponding CACHE ID and is identical to the "torrents" array in format. The "torrentm" array contains a list of hashes for

ID and is identical to the "torrents" array in format. The "torrentm" array contains a list of hashes for torrent jobs that have been removed since the corresponding CACHE ID. A new CACHE ID will be given in torrentc and this can be used for the next list request.

Files List

To get the list of files in a torrent job, request this URI: http://[IP]:[PORT]/gui/?action=getfiles&hash=[TORRENT HASH] . This will return the following:

```
{
        "build": BUILD NUMBER (integer),
        "files": [
        HASH (string),
        [
        [
        FILE NAME (string),
        FILE SIZE (integer in bytes),
        DOWNLOADED (integer in bytes),
        PRIORITY* (integer)     ],
        ...
        ]
        ]
}
```

* PRIORITY: This is an integer value that indicates the file's priority. The following is a list of the possible PRIORITY values and what each corresponds to:

- 0 = Don't Download
- 1 = Low Priority
- 2 = Normal Priority
- 3 = High Priority

This command accepts multiple hashes. It will return multiple "files" key/value pairs.

Torrent Job Properties

To get a list of the various properties for a torrent job, request http://[IP]:[PORT]/gui/?action=getprops&hash=[TORRENT HASH] . This will return the following:

```
{
        "build": BUILD NUMBER (integer),
        "props": [
        {
```

```
        {
        "hash": HASH (string),
        "trackers": TRACKERS* (string),
        "ulrate": UPLOAD LIMIT (integer in bytes per second),
        "dlrate": DOWNLOAD LIMIT (integer in bytes per second),
        "superseed": INITIAL SEEDING** (integer),
        "dht": USE DHT** (integer),
        "pex": USE PEX** (integer),
        "seed_override": OVERRIDE QUEUEING** (integer),
        "seed_ratio": SEED RATIO (integer in per mils),
        "seed_time": SEEDING TIME*** (integer in seconds),
        "ulslots": UPLOAD SLOTS (integer)       }
        ]
}
```

* TRACKER: This is a list of the trackers used by the torrent job. Each newline is represented by a carriage return followed by a newline (\r\n).

** INITIAL SEEDING/USE DHT/USE PEX/OVERRIDE QUEUEING: These options are all integer values that indicate their respective states. The following is a list of the possible values and what each corresponds to:

- -1 = Not allowed
- 0 = Disabled
- 1 = Enabled

*** SEEDING TIME: This is an integer representing the minimum amount of time (in seconds) that µTorrent should continue to seed after it has finished downloading the torrent. A value of 0 (zero) means no minimum seeding time.

To change the properties for a torrent, the following URI can be used: http://[IP]:[PORT]/gui/?action=setprops&hash=[TORRENT HASH]&s=[PROPERTY]&v=[VALUE]

Multiple properties can be changed at once by appending more s= and v= pairs. Multiple torrent jobs can be modified in a single request by appending another hash= value along with its s= and v= pairs. Any following s= and v= pairs will modify the properties of the last specified hash.

For example, to set an upload rate limit of 10 KiB/s and a download rate of 20 KiB/s for [TORRENT HASH 1], while simultaneously setting 4 upload slots for [TORRENT HASH 2], request the following URI: http://[IP]:[PORT]/gui/?action=setprops&hash=[TORRENT HASH 1]&s=ulrate&v=10240&s=dlrate&v=20480&hash=[TORRENT HASH 2]&s=ulslots&v=4

Actions

This section contains a list of all the other possible actions supported by the API.

This section contains a list of all the other possible actions supported by the API.

http://[IP]:[PORT]/gui/?action=start&hash=[TORRENT HASH]

This action tells µTorrent to start the specified torrent job(s). Multiple hashes may be specified to act on multiple torrent jobs.

http://[IP]:[PORT]/gui/?action=stop&hash=[TORRENT HASH]

This action tells µTorrent to stop the specified torrent job(s). Multiple hashes may be specified to act on multiple torrent jobs.

http://[IP]:[PORT]/gui/?action=pause&hash=[TORRENT HASH]

This action tells µTorrent to pause the specified torrent job(s). Multiple hashes may be specified to act on multiple torrent jobs.

http://[IP]:[PORT]/gui/?action=forcestart&hash=[TORRENT HASH]

This action tells µTorrent to force the specified torrent job(s) to start. Multiple hashes may be specified to act on multiple torrent jobs.

http://[IP]:[PORT]/gui/?action=unpause&hash=[TORRENT HASH]

This action tells µTorrent to unpause the specified torrent job(s). Multiple hashes may be specified to act on multiple torrent jobs.

http://[IP]:[PORT]/gui/?action=recheck&hash=[TORRENT HASH]

This action tells µTorrent to recheck the torrent contents for the specified torrent job(s). Multiple hashes may be specified to act on multiple torrent jobs.

http://[IP]:[PORT]/gui/?action=remove&hash=[TORRENT HASH]

This action removes the specified torrent job(s) from the torrent jobs list. Multiple hashes may be specified to act on multiple torrent jobs. This action respects the option "Move to trash if possible".

http://[IP]:[PORT]/gui/?action=removedata&hash=[TORRENT HASH]

This action removes the specified torrent job(s) from the torrent jobs list and removes the corresponding torrent contents (data) from disk. Multiple hashes may be specified to act on multiple torrent jobs. This action respects the option "Move to trash if possible".

http://[IP]:[PORT]/gui/?action=setprio&hash=[TORRENT HASH]&p=[PRIORITY]&f=[FILE INDEX]

This action sets the priority for the specified file(s) in the torrent job. The possible priority levels are the values returned by "getfiles". A file is specified using the zero-based index of the file in the inside the list returned by "getfiles". Only one priority level may be specified on each call to this action, but multiple files may be specified.

http://[IP]:[PORT]/gui/?action=add-url&s=[TORRENT URL]

http://[IP]:[PORT]/gui/?action=add-url&s=[TORRENT URL]

This action adds a torrent job from the given URL. For servers that require cookies, cookies can be sent with the :COOKIE: method (see here). The string must be URL-encoded.

http://[IP]:[PORT]/gui/?action=add-file

This action is different from the other actions in that it uses HTTP POST instead of HTTP GET to submit data to µTorrent. The HTTP form must use an enctype of "multipart/form-data" and have an input field of type "file" with name "torrent_file" that stores the local path to the file to upload to µTorrent.

Limitations

It is not possible to specify a path to save a torrent's data. It is possible to set the default download directory with getsettings, but not for each individual torrent. This will be rectified in a later release.

It is not possible to rename or relocate individual files in a torrent job.

RSS is not implemented in the API.

Credits

Thanks go out to Ultima for documenting the vast majority of the API on the forums, where most of the info on this page was taken.

Like

@uTorrent

µTorrent Blog

Introducing BTC: A Torrenting Shortcut

View Post

Co

Get µTorrent
Version 3.3.2 for Windows

Get Started
Find Torrents
Skins

Community
Forums
Idea Bank

Help
FAQ
Forums

Other Platforms & Languages

Developers

Twitter

Facebook

Videos

About Us Blog Press Privacy EULA Terms