gmusicapi Documentation

Release 11.0.0

Simon Weber

Dec 10, 2017

Contents

This library allows control of Google Music with Python.

```
from gmusicapi import Mobileclient

api = Mobileclient()
api.login('user@gmail.com', 'my-password', Mobileclient.FROM_MAC_ADDRESS)
# => True

library = api.get_all_songs()
sweet_track_ids = [track['id'] for track in library
                   if track['artist'] == 'The Cat Empire']

playlist_id = api.create_playlist('Rad muzak')
api.add_songs_to_playlist(playlist_id, sweet_track_ids)
```

This project is not supported nor endorsed by Google. Use common sense (protocol compliance, reasonable load, etc) and don't ruin the fun for everyone else.

CHAPTER 1

Features

All major functionality is supported:

- Library management: list, create, delete, and modify songs and playlists
- Streaming and single-song downloading
- Music Manager uploading/scan-and-match and library downloading
- Most All Access features

See the changelog for changes by version.

CHAPTER 2

Using gmusicapi

2.1 Usage

2.1.1 Installation

Use pip: `$ pip install gmusicapi`. This will grab all the source dependencies. Avoid using `easy_install`.

If you're upgrading from a date-versioned release (eg `2013.03.04`), do `$ pip uninstall gmusicapi; pip install gmusicapi` instead.

If you're going to be uploading music, you'll likely need avconv or ffmpeg installed and in your system path, along with at least libmp3lame.

- Linux
  - Use your distro's package manager: e.g `$ sudo apt-get install libav-tools libavcodec-extra-53` (ffmpeg requires extra steps on Debian/Ubuntu).
  - Download pre-built binaries of avconv or ffmpeg and edit your path to include the directory that contains avconv/ffmpeg.
- Mac
  - Use Homebrew to install libav (avconv) or ffmpeg.
- Windows
  - Download pre-built binaries of avconv or ffmpeg and edit your path to include the directory that contains avconv.exe/ffmpeg.exe.
- Google App Engine
  - See this thread for instructions.

The only time avconv or ffmpeg is not required is when uploading mp3s without scan-and-match enabled.

If you need to install avconv/ffmpeg from source, be sure to use `$ ./configure --enable-gpl --enable-nonfree --enable-libmp3lame`.

2.1.2 Quickstart

If you're not going to be uploading music, you'll likely want to use the *Mobileclient*: it supports streaming and library management. It requires plaintext auth, so your code might look something like:

```
from gmusicapi import Mobileclient

api = Mobileclient()
logged_in = api.login('user@gmail.com', 'my-password', Mobileclient.FROM_MAC_ADDRESS)
# logged_in is True if login was successful
```

Note that 2-factor users will need to setup and provide an app-specific password.

If you're going to upload Music, you want the *Musicmanager*. It uses OAuth2 and does not require plaintext credentials.

Instead, you'll need to authorize your account *once* before logging in. The easiest way is to run:

```
from gmusicapi import Musicmanager

mm = Musicmanager()
mm.perform_oauth()
```

If successful, this will save your credentials to disk. Then, future runs can start with:

```
from gmusicapi import Musicmanager

mm = Musicmanager()
mm.login()

# mm.upload('foo.mp3')
# ...
```

If you need both library management and uploading, just create multiple client instances.

There is also the *Webclient*, which is a mostly-deprecated interface. It is not tested nor well supported. Use *Mobileclient* or *Musicmanager* if possible.

The reference section has complete information on all clients:

Client Interfaces

gmusicapi currently has three main interfaces: one for the music.google.com webclient, one for the Android App, and one for the Music Manager. The big differences are:

- *Webclient* development has mostly ceased, with the *Mobileclient* superceding it. It is not tested nor well supported.
- *Musicmanager* is used for uploading and downloading, while *Mobileclient* supports everything but uploading.
- *Webclient*/*Mobileclient* require a plaintext email and password to login, while *Musicmanager* uses OAuth2.
- *Mobileclient* supports streaming but requires that the Google Play Music app has been installed and run before use.

Webclient Interface

WARNING Webclient functionality is not tested nor well supported. Use *Mobileclient* or *Musicmanager* if possible.

class gmusicapi.clients.**Webclient**(*debug_logging=True*, *validate=True*, *verify_ssl=True*)

Allows library management and streaming by posing as the music.google.com webclient.

Uploading is not supported by this client (use the *Musicmanager* to upload).

Any methods in this class that are duplicated by the *Mobileclient* are deprecated, and will generate a warning at runtime.

The following methods are *not* deprecated:

- *get_shared_playlist_info()*
- *get_song_download_info()*
- *get_stream_urls()*
- *get_stream_audio()*
- *report_incorrect_match()*
- *upload_album_art()*

Setup and login

Webclient.**__init__**(*debug_logging=True*, *validate=True*, *verify_ssl=True*)

Parameters

- **debug_logging** – each Client has a `logger` member. The logger is named `gmusicapi.<client class><client number>` and will propogate to the `gmusicapi` root logger.

  If this param is `True`, handlers will be configured to send this client's debug log output to disk, with warnings and above printed to stderr. Appdirs `user_log_dir` is used by default. Users can run:

  ```
  from gmusicapi.utils import utils
  print utils.log_filepath
  ```

  to see the exact location on their system.

  If `False`, no handlers will be configured; users must create their own handlers.

  Completely ignoring logging is dangerous and not recommended. The Google Music protocol can change at any time; if something were to go wrong, the logs would be necessary for recovery.

- **validate** – if False, do not validate server responses against known schemas. This helps to catch protocol changes, but requires significant cpu work.

  This arg is stored as `self.validate` and can be safely modified at runtime.

- **verify_ssl** – if False, exceptions will not be raised if there are problems verifying SSL certificates. Be wary of using this option; it's almost always better to fix the machine's SSL configuration than to ignore errors.

Webclient.**login**(*email*, *password*)
Authenticates the webclient. Returns `True` on success, `False` on failure.

Parameters

- **email** – eg `'test@gmail.com'` or just `'test'`.
- **password** – the account's password. This is not stored locally, and is sent securely over SSL. App-specific passwords are not supported on the webclient.

Users who don't use two-factor auth will likely need to enable less secure login. If this is needed, a warning will be logged during login (which will print to stderr in the default logging configuration).

Webclient.**logout**()
Forgets local authentication and cached properties in this Api instance. Returns `True` on success.

Song downloading and streaming

Webclient.**get_song_download_info**(*song_id*)
Returns a tuple: `('<url>', <download count>)`.

Parameters **song_id** – a single song id.

`url` will be `None` if the download limit is exceeded.

GM allows 2 downloads per song. The download count may not always be accurate, and the 2 download limit seems to be loosely enforced.

This call alone does not count towards a download - the count is incremented when `url` is retrieved.

Webclient.**get_stream_audio**(*song_id*, *use_range_header=None*)
Returns a bytestring containing mp3 audio for this song.

Parameters

- **song_id** – a single song id
- **use_range_header** – in some cases, an HTTP range header can be used to save some bandwidth. However, there's no guarantee that the server will respect it, meaning that the client may get back an unexpected response when using it.

  There are three possible values for this argument:

  - None: (default) send header; fix response locally on problems
  - True: send header; raise IOError on problems
  - False: do not send header

Webclient.**get_stream_urls**(*song_id*)
Returns a list of urls that point to a streamable version of this song.

If you just need the audio and are ok with gmusicapi doing the download, consider using *get_stream_audio()* instead. This abstracts away the differences between different kinds of tracks:

- normal tracks return a single url
- All Access tracks return multiple urls, which must be combined

Parameters **song_id** – a single song id.

While acquiring the urls requires authentication, retreiving the contents does not.

However, there are limitations on how the stream urls can be used:

- the urls expire after a minute
- only one IP can be streaming music at once. Other attempts will get an http 403 with `X-Rejected-Reason: ANOTHER_STREAM_BEING_PLAYED`.

This is only intended for streaming. The streamed audio does not contain metadata. Use *get_song_download_info()* or *Musicmanager.download_song* to download files with metadata.

`Webclient.`**`report_incorrect_match`**(*song_ids*)

Equivalent to the 'Fix Incorrect Match' button, this requests re-uploading of songs. Returns the song_ids provided.

Parameters **song_ids** – a list of song ids to report, or a single song id.

Note that if you uploaded a song through gmusicapi, it won't be reuploaded automatically - this currently only works for songs uploaded with the Music Manager. See issue #89.

This should only be used on matched tracks (`song['type'] == 6`).

Song manipulation

`Webclient.`**`upload_album_art`**(*song_ids*, *image_filepath*)

Uploads an image and sets it as the album art for songs. Returns a url to the image on Google's servers.

Parameters

- **song_ids** – a list of song ids, or a single song id.
- **image_filepath** – filepath of the art to use. jpg and png are known to work.

This function will *always* upload the provided image, even if it's already uploaded. If the art is already uploaded and set for another song, copy over the value of the `'albumArtUrl'` key using *Mobileclient.change_song_metadata()* instead.

`Webclient.`**`delete_songs`**(*song_ids*)

Deprecated: prefer *Mobileclient.delete_songs()*.

Deletes songs from the entire library. Returns a list of deleted song ids.

Parameters **song_ids** – a list of song ids, or a single song id.

Playlist manipulation

`Webclient.`**`create_playlist`**(*name*, *description=None*, *public=False*)

Creates a playlist and returns its id.

Parameters

- **name** – the name of the playlist.
- **description** – (optional) the description of the playlist.
- **public** – if True and the user has All Access, create a shared playlist.

`Webclient.`**`add_songs_to_playlist`**(*playlist_id*, *song_ids*)

Deprecated: prefer *Mobileclient.add_songs_to_playlist()*.

Appends songs to a playlist. Returns a list of (song id, playlistEntryId) tuples that were added.

Parameters

- **playlist_id** – id of the playlist to add to.

- **song_ids** – a list of song ids, or a single song id.

Playlists have a maximum size of 1000 songs.

Webclient.**remove_songs_from_playlist**(*playlist_id*, *sids_to_match*)

Deprecated: prefer *Mobileclient.remove_entries_from_playlist()*.

Removes all copies of the given song ids from a playlist. Returns a list of removed (sid, eid) pairs.

Parameters

- **playlist_id** – id of the playlist to remove songs from.
- **sids_to_match** – a list of song ids to match, or a single song id.

This does *not always* the inverse of a call to *add_songs_to_playlist()*, since multiple copies of the same song are removed.

Webclient.**get_shared_playlist_info**(*share_token*)

Returns a dictionary with four keys: author, description, num_tracks, and title.

Parameters **share_token** – from `playlist['shareToken']`, or a playlist share url (`https://play.google.com/music/playlist/<token>`).

Note that tokens from urls will need to be url-decoded, eg `AM...%3D%3D` becomes `AM...==`.

Other

Webclient.**get_registered_devices**()

Returns a list of dictionaries representing devices associated with the account.

Performing the *Musicmanager* OAuth flow will register a device of type 1.

Installing the Google Music app on an android or ios device and logging into it will register a device of type 2 or 3, which is used for streaming with the *Mobileclient*.

Here is an example response:

```
[
  {
    u'deviceType': 1,  # laptop/desktop
    u'id': u'00:11:22:33:AA:BB',
    u'lastAccessedFormatted': u'May 24, 2015',
    u'lastAccessedTimeMillis': 1432468588200,  # utc-millisecond
    u'lastEventTimeMillis': 1434211605335,
    u'name': u'my computer'},
  },
  {
    u'deviceType': 2,  # android device
    u'carrier': u'Google',
    u'id': u'0x00112233aabbccdd',  # remove 0x when streaming
    u'lastAccessedFormatted': u'September 19, 2015',
    u'lastAccessedTimeMillis': 1442706069906,
    u'lastEventTimeMillis': 1435271137193,
    u'manufacturer': u'Asus',
    u'model': u'Nexus 7',
    u'name': u'my nexus 7'
  },
  {
    u'deviceType': 3,  # ios device
    u'id': u'ios:01234567-0123-0123-0123-0123456789AB',
```

```
    u'lastAccessedFormatted': u'June 25, 2015',
    u'lastAccessedTimeMillis': 1435271588780,
    u'lastEventTimeMillis': 1435271442417,
    u'name': u'my iphone'
  }
]
```

Mobileclient Interface

class gmusicapi.clients.**Mobileclient**(*debug_logging=True*, *validate=True*, *verify_ssl=True*)
Allows library management and streaming by posing as the googleapis.com mobile clients.

Uploading is not supported by this client (use the *Musicmanager* to upload).

Setup and login

Mobileclient.**__init__**(*debug_logging=True*, *validate=True*, *verify_ssl=True*)

Parameters

- **debug_logging** – each Client has a `logger` member. The logger is named `gmusicapi.<client class><client number>` and will propogate to the `gmusicapi` root logger.

  If this param is `True`, handlers will be configured to send this client's debug log output to disk, with warnings and above printed to stderr. Appdirs `user_log_dir` is used by default. Users can run:

  ```
  from gmusicapi.utils import utils
  print utils.log_filepath
  ```

  to see the exact location on their system.

  If `False`, no handlers will be configured; users must create their own handlers.

  Completely ignoring logging is dangerous and not recommended. The Google Music protocol can change at any time; if something were to go wrong, the logs would be necessary for recovery.

- **validate** – if False, do not validate server responses against known schemas. This helps to catch protocol changes, but requires significant cpu work.

  This arg is stored as `self.validate` and can be safely modified at runtime.

- **verify_ssl** – if False, exceptions will not be raised if there are problems verifying SSL certificates. Be wary of using this option; it's almost always better to fix the machine's SSL configuration than to ignore errors.

Mobileclient.**login**(*email*, *password*, *android_id*, *locale=u'en_US'*)
Authenticates the Mobileclient. Returns `True` on success, `False` on failure.

Parameters

- **email** – eg `'test@gmail.com'` or just `'test'`.
- **password** – password or app-specific password for 2-factor users. This is not stored locally, and is sent securely over SSL.

- **android_id** – 16 hex digits, eg `'1234567890abcdef'`.

  Pass Mobileclient.FROM_MAC_ADDRESS instead to attempt to use this machine's MAC address as an android id. **Use this at your own risk**: the id will be a non-standard 12 characters, but appears to work fine in testing. If a valid MAC address cannot be determined on this machine (which is often the case when running on a VPS), raise OSError.

- **locale** – ICU locale used to localize certain responses. This must be a locale supported by Android. Defaults to `'en_US'`.

`Mobileclient.logout()`
Forgets local authentication and cached properties in this Api instance. Returns `True` on success.

`Mobileclient.is_authenticated()`
Returns `True` if the Api can make an authenticated request.

`Mobileclient.locale`
The locale of the Mobileclient session used to localize some responses.

Should be an ICU locale supported by Android.

Set on authentication with *`login()`* but can be changed at any time.

Account Management

`Mobileclient.is_subscribed`
Returns the subscription status of the Google Music account.

Result is cached with a TTL of 10 minutes. To get live status before the TTL is up, delete the `is_subscribed` property of the Mobileclient instance.

```
>>> mc = Mobileclient()
>>> mc.is_subscribed  # Live status.
>>> mc.is_subscribed  # Cached status.
>>> del mc.is_subscribed  # Delete is_subscribed property.
>>> mc.is_subscribed  # Live status.
```

`Mobileclient.get_registered_devices()`
Returns a list of dictionaries representing devices associated with the account.

Performing the *`Musicmanager`* OAuth flow will register a device of type `'DESKTOP_APP'`.

Installing the Android or iOS Google Music app and logging into it will register a device of type `'ANDROID'` or `'IOS'` respectively, which is required for streaming with the *`Mobileclient`*.

Here is an example response:

```
[
  {
    u'kind':               u'sj#devicemanagementinfo',
    u'friendlyName':       u'my-hostname',
    u'id':                 u'AA:BB:CC:11:22:33',
    u'lastAccessedTimeMs': u'1394138679694',
    u'type':               u'DESKTOP_APP'
  },
  {
    u"kind":               u"sj#devicemanagementinfo",
    u'friendlyName':       u'Nexus 7',
    u'id':                 u'0x00112233aabbccdd',  # remove 0x when streaming
    u'lastAccessedTimeMs': u'1344808742774',
```

```
    u'type':                u'ANDROID'
    u'smartPhone':          True
  },
  {
    u"kind":                u"sj#devicemanagementinfo",
    u'friendlyName':        u'iPhone 6',
    u'id':                  u'ios:01234567-0123-0123-0123-0123456789AB',
    u'lastAccessedTimeMs': 1394138679694,
    u'type':                u'IOS'
    u'smartPhone':          True
  }
  {
    u'kind':                u'sj#devicemanagementinfo',
    u'friendlyName':        u'Google Play Music for Chrome on Windows',
    u'id':                  u'rt2qfkh0qjhos4bxrgc0oae...',  # 64 characters,␣
↪alphanumeric
    u'lastAccessedTimeMs': u'1425602805052',
    u'type':                u'DESKTOP_APP'
  },
]
```

Mobileclient.**deauthorize_device**(*device_id*)

Deauthorize a registered device.

Returns `True` on success, `False` on failure.

Parameters **device_id** – A mobile device id as a string. Android ids are 16 characters with '0x' prepended, iOS ids are uuids with 'ios:' prepended, while desktop ids are in the form of a MAC address.

Providing an invalid or unregistered device id will result in a 400 HTTP error.

Google limits the number of device deauthorizations to 4 per year. Attempts to deauthorize a device when that limit is reached results in a 403 HTTP error with: `X-Rejected-Reason: TOO_MANY_DEAUTHORIZATIONS`.

Songs

Songs are uniquely referred to within a library with a track id in uuid format.

Store tracks also have track ids, but they are in a different format than library track ids. `song_id.startswith('T')` is always `True` for store track ids and `False` for library track ids.

Adding a store track to a library will yield a normal song id.

Store track ids can be used in most places that normal song ids can (e.g. playlist addition or streaming). Note that sometimes they are stored under the `'nid'` key, not the `'id'` key.

Mobileclient.**get_all_songs**(*incremental=False*, *include_deleted=None*, *updated_after=None*)

Returns a list of dictionaries that each represent a song.

Parameters

- **incremental** – if True, return a generator that yields lists of at most 1000 tracks as they are retrieved from the server. This can be useful for presenting a loading bar to a user.
- **include_deleted** – ignored. Will be removed in a future release.
- **updated_after** – a datetime.datetime; defaults to unix epoch. If provided, deleted songs may be returned.

Here is an example song dictionary:


```
{
   'comment':'',
   'rating':'0',
   'albumArtRef':[
     {
       'url': 'http://lh6.ggpht.com/...'
     }
   ],
   'artistId':[
     'Aod62yyj3u3xsjtooghh2glwsdi'
   ],
   'composer':'',
   'year':2011,
   'creationTimestamp':'1330879409467830',
   'id':'5924d75a-931c-30ed-8790-f7fce8943c85',
   'album':'Heritage ',
   'totalDiscCount':0,
   'title':'Haxprocess',
   'recentTimestamp':'1372040508935000',
   'albumArtist':'',
   'trackNumber':6,
   'discNumber':0,
   'deleted':False,
   'storeId':'Txsffypukmmeg3iwl3w5a5s3vzy',
   'nid':'Txsffypukmmeg3iwl3w5a5s3vzy',
   'totalTrackCount':10,
   'estimatedSize':'17229205',
   'albumId':'Bdkf6ywxmrhflvtasnayxlkgpcm',
   'beatsPerMinute':0,
   'genre':'Progressive Metal',
   'playCount':7,
   'artistArtRef':[
     {
       'url': 'http://lh3.ggpht.com/...'
     }
   ],
   'kind':'sj#track',
   'artist':'Opeth',
   'lastModifiedTimestamp':'1330881158830924',
   'clientId':'+eGFGTbiyMktbPuvB5MfsA',
   'durationMillis':'418000'
 }
```


`Mobileclient.`**`get_stream_url`**(*song_id*, *device_id=None*, *quality='hi'*)

Returns a url that will point to an mp3 file.

Parameters

- **`song_id`** – A single song id. This can be `'storeId'` from a store song, `'id'` from an uploaded song, or `'trackId'` from a playlist entry.
- **`device_id`** – (optional) defaults to `android_id` from login.

  Otherwise, provide a mobile device id as a string. Android device ids are 16 characters, while iOS ids are uuids with 'ios:' prepended.

  If you have already used Google Music on a mobile device, *`Mobileclient.get_registered_devices`* will provide at least one working id. Omit `'0x'` from

the start of the string if present.

Registered computer ids (a MAC address) will not be accepted and will 403.

Providing an unregistered mobile device id will register it to your account, subject to Google's device limits. **Registering a device id that you do not own is likely a violation of the TOS.**

- **quality** – (optional) stream bits per second quality One of three possible values, hi: 320kbps, med: 160kbps, low: 128kbps. The default is hi

When handling the resulting url, keep in mind that:

- you will likely need to handle redirects
- the url expires after a minute
- only one IP can be streaming music at once. This can result in an http 403 with `X-Rejected-Reason: ANOTHER_STREAM_BEING_PLAYED`.

The file will not contain metadata. Use *Webclient.get_song_download_info* or *Musicmanager.download_song* to download files with metadata.

Mobileclient.**rate_songs**(*songs*, *rating*)
Rate library or store songs.

Returns rated song ids.

Parameters

- **songs** – a list of song dictionaries or a single song dictionary. required keys: 'id' for library songs or 'nid' and 'trackType' for store songs.
- **rating** – set to `'0'` (no thumb), `'1'` (down thumb), or `'5'` (up thumb).

Mobileclient.**change_song_metadata**(*songs*)
Changes the metadata of tracks. Returns a list of the song ids changed.

Parameters **songs** – a list of song dictionaries or a single song dictionary.

Currently, only the `rating` key can be changed. Set it to `'0'` (no thumb), `'1'` (down thumb), or `'5'` (up thumb).

You can also use this to rate store tracks that aren't in your library, eg:

```
song = mc.get_track_info('<some store track id>')
song['rating'] = '5'
mc.change_song_metadata(song)
```

Mobileclient.**delete_songs**(*library_song_ids*)
Deletes songs from the library. Returns a list of deleted song ids.

Parameters **song_ids** – a list of song ids, or a single song id.

Mobileclient.**get_promoted_songs**()
Returns a list of dictionaries that each represent a track.

Only store tracks will be returned.

Promoted tracks are determined in an unknown fashion, but positively-rated library tracks are common.

See *get_track_info()* for the format of a track dictionary.

Mobileclient.**increment_song_playcount**(*song_id*, *plays=1*, *playtime=None*)
Increments a song's playcount and returns its song id.

Params song_id a song id. Providing the id of a store track that has been added to the library will *not* increment the corresponding library song's playcount. To do this, use the 'id' field (which looks like a uuid and doesn't begin with 'T'), not the 'nid' field.

Params plays (optional) positive number of plays to increment by. The default is 1.

Params playtime (optional) a datetime.datetime of the time the song was played. It will default to the time of the call.

Mobileclient.**add_store_track**(*store_song_id*)
Adds a store track to the library

Returns the library track id of added store track.

Parameters **store_song_id** – store song id

Mobileclient.**add_store_tracks**(*store_song_ids*)
Add store tracks to the library

Returns a list of the library track ids of added store tracks.

Parameters **store_song_ids** – a list of store song ids or a single store song id

Mobileclient.**get_station_track_stream_url**(*song_id*, *wentry_id*, *session_token*, *quality=u'hi'*)
Returns a url that will point to an mp3 file.

This is only for use by free accounts, and requires a call to *get_station_info()* first to provide *wentry_id* and *session_token*. Subscribers should instead use *get_stream_url()*.

Parameters

- **song_id** – a single song id
- **wentry_id** – a free radio station track entry id (*wentryid* from *get_station_info()*)
- **session_token** – a free radio station session token (*sessionToken* from *get_station_info()*)
- **quality** – (optional) stream bits per second quality One of three possible values, hi: 320kbps, med: 160kbps, low: 128kbps. The default is hi

Playlists

Like songs, playlists have unique ids within a library. However, their names do not need to be unique.

The tracks making up a playlist are referred to as 'playlist entries', and have unique entry ids within the entire library (not just their containing playlist).

Mobileclient.**get_all_playlists**(*incremental=False*, *include_deleted=None*, *updated_after=None*)
Returns a list of dictionaries that each represent a playlist.

Parameters

- **incremental** – if True, return a generator that yields lists of at most 1000 playlists as they are retrieved from the server. This can be useful for presenting a loading bar to a user.
- **include_deleted** – ignored. Will be removed in a future release.
- **updated_after** – a datetime.datetime; defaults to unix epoch If provided, deleted playlists may be returned.

Here is an example playlist dictionary:

```
{
     # can also be SHARED (public/subscribed to), MAGIC or omitted
    'type': 'USER_GENERATED',

    'kind': 'sj#playlist',
    'name': 'Something Mix',
    'deleted': False,
    'lastModifiedTimestamp': '1325458766483033',
    'recentTimestamp': '1325458766479000',
    'shareToken': '<long string>',
    'ownerProfilePhotoUrl': 'http://lh3.googleusercontent.com/...',
    'ownerName': 'Simon Weber',
    'accessControlled': False,  # has to do with shared playlists
    'creationTimestamp': '1325285553626172',
    'id': '3d72c9b5-baad-4ff7-815d-cdef717e5d61'
}
```

Mobileclient.**get_all_user_playlist_contents**()

Retrieves the contents of *all* user-created playlists – the Mobileclient does not support retrieving only the contents of one playlist.

This will not return results for public playlists that the user is subscribed to; use *get_shared_playlist_contents()* instead.

The same structure as *get_all_playlists()* will be returned, but with the addition of a `'tracks'` key in each dict set to a list of properly-ordered playlist entry dicts.

Here is an example playlist entry for an individual track:

```
{
    'kind': 'sj#playlistEntry',
    'deleted': False,
    'trackId': '2bb0ab1c-ce1a-3c0f-9217-a06da207b7a7',
    'lastModifiedTimestamp': '1325285553655027',
    'playlistId': '3d72c9b5-baad-4ff7-815d-cdef717e5d61',
    'absolutePosition': '01729382256910287871',  # denotes playlist ordering
    'source': '1',  # '2' if hosted on Google Music, '1' otherwise (see below)
    'creationTimestamp': '1325285553655027',
    'id': 'c9f1aff5-f93d-4b98-b13a-429cc7972fea' ## see below
}
```

If a user uploads local music to Google Music using the Music Manager, Google will attempt to match each uploaded track to a track already hosted on its servers. If a match is found for a track, the playlist entry key `'source'` has the value `'2'`, and the entry will have a key `'track'` with a value that is a dict of track metadata (title, artist, etc).

If a track is not hosted on Google Music, then the playlist entry key `'source'` has the value `'1'`, and may not have a `'track'` key (e.g., for an MP3 without ID3 tags). In this case, the key `'trackId'` corresponds to the column `ServerId` in the table `XFILES` in Music Manager's local SQLite database (stored, e.g., at ~/Library/ApplicationSupport/Google/MusicManager/ServerDatabase.db on OS X). Among other things, the SQLite database exposes the track's local file path, and Music Manager's imputed metadata.

(Note that the above behavior is documented for the Music Manager set to sync from local Folders, and may differ if it instead syncs from iTunes.)

Mobileclient.**get_shared_playlist_contents**(*share_token*)

Retrieves the contents of a public playlist.

Parameters **share_token** – from `playlist['shareToken']`, or a playlist share url

(`https://play.google.com/music/playlist/<token>`).

Note that tokens from urls will need to be url-decoded, eg `AM...%3D%3D` becomes `AM...==`.

For example, to retrieve the contents of a playlist that the user is subscribed to:

```
subscribed_to = [p for p in mc.get_all_playlists() if p.get('type') == 'SHARED']
share_tok = subscribed_to[0]['shareToken']
tracks = mc.get_shared_playlist_contents(share_tok)
```

The user need not be subscribed to a playlist to list its tracks.

Returns a list of playlist entries with structure the same as those returned by *get_all_user_playlist_contents()*, but without the `'clientId'` or `'playlistId'` keys.

`Mobileclient.`**`create_playlist`**(*name*, *description=None*, *public=False*)
Creates a new empty playlist and returns its id.

Parameters

- **`name`** – the desired title. Creating multiple playlists with the same name is allowed.
- **`description`** – (optional) the desired description
- **`public`** – (optional) if True and the user has a subscription, share playlist.

`Mobileclient.`**`delete_playlist`**(*playlist_id*)
Deletes a playlist and returns its id.

Parameters **`playlist_id`** – the id to delete.

`Mobileclient.`**`edit_playlist`**(*playlist_id*, *new_name=None*, *new_description=None*, *public=None*)
Changes the name of a playlist and returns its id.

Parameters

- **`playlist_id`** – the id of the playlist
- **`new_name`** – (optional) desired title
- **`new_description`** – (optional) desired description
- **`public`** – (optional) if True and the user has a subscription, share playlist.

`Mobileclient.`**`add_songs_to_playlist`**(*playlist_id*, *song_ids*)
Appends songs to the end of a playlist. Returns a list of playlist entry ids that were added.

Parameters

- **`playlist_id`** – the id of the playlist to add to.
- **`song_ids`** – a list of song ids, or a single song id. These can be `'storeId'` from a store song, `'id'` from an uploaded song, or `'trackId'` from a playlist entry.

Playlists have a maximum size of 1000 songs. Calls may fail before that point (presumably) due to an error on Google's end (see #239).

`Mobileclient.`**`remove_entries_from_playlist`**(*entry_ids*)
Removes specific entries from a playlist. Returns a list of entry ids that were removed.

Parameters **`entry_ids`** – a list of entry ids, or a single entry id.

`Mobileclient.`**`reorder_playlist_entry`**(*entry*, *to_follow_entry=None*, *to_precede_entry=None*)
Reorders a single entry in a playlist and returns its id.

Read `reorder_playlist_entry(foo, bar, gaz)` as "reorder playlist entry *foo* to follow entry *bar* and precede entry *gaz*."

Parameters

- **entry** – the playlist entry to move.
- **to_follow_entry** – the playlist entry that will come before *entry* in the resulting playlist, or None if *entry* is to be the first entry in the playlist.
- **to_precede_entry** – the playlist entry that will come after *entry* in the resulting playlist or None if *entry* is to be the last entry in the playlist.

`reorder_playlist_entry(foo)` is invalid and will raise ValueError; provide at least one of *to_follow_entry* or *to_precede_entry*.

Leaving *to_follow_entry* or *to_precede_entry* as None when *entry* is not to be the first or last entry in the playlist is undefined.

All params are dicts returned by *get_all_user_playlist_contents()* or *get_shared_playlist_contents()*.

Radio Stations

Radio Stations are available for free in the US only. A subscription is required in other countries.

`Mobileclient.`**`get_all_stations`**(*incremental=False*, *include_deleted=None*, *updated_after=None*)

Retrieve all library stations.

Returns a list of dictionaries that each represent a radio station.

Parameters

- **incremental** – if True, return a generator that yields lists of at most 1000 stations as they are retrieved from the server. This can be useful for presenting a loading bar to a user.
- **include_deleted** – ignored. Will be removed in a future release.
- **updated_after** – a datetime.datetime; defaults to unix epoch

Here is an example station dictionary:

```
{
    'imageUrl': 'http://lh6.ggpht.com/...',
    'kind': 'sj#radioStation',
    'name': 'station',
    'deleted': False,
    'lastModifiedTimestamp': '1370796487455005',
    'recentTimestamp': '1370796487454000',
    'clientId': 'c2639bf4-af24-4e4f-ab37-855fc89d15a1',
    'seed':
    {
        'kind': 'sj#radioSeed',
        'trackLockerId': '7df3aadd-9a18-3dc1-b92e-a7cf7619da7e'
        # possible keys:
        #  albumId, artistId, genreId, trackId, trackLockerId
    },
    'id': '69f1bfce-308a-313e-9ed2-e50abe33a25d'
},
```

`Mobileclient.`**`create_station`**(*name*, *track_id=None*, *artist_id=None*, *album_id=None*, *genre_id=None*, *playlist_token=None*, *curated_station_id=None*)

Creates a radio station and returns its id.

Parameters

- **name** – the name of the station to create
- ***_id** – the id of an item to seed the station from.
- **playlist_token** – The shareToken of a playlist to seed the station from.

Exactly one of the id/token params must be provided, or ValueError will be raised.

Mobileclient.**delete_stations**(*station_ids*)
Deletes radio stations and returns their ids.

Parameters **station_ids** – a single id, or a list of ids to delete

Mobileclient.**get_station_tracks**(*station_id*, *num_tracks=25*, *recently_played_ids=None*)
Returns a list of dictionaries that each represent a track.

Each call performs a separate sampling (with replacement?) from all possible tracks for the station.

Nonexistent stations will return an empty list.

Parameters

- **station_id** – the id of a radio station to retrieve tracks from. Use the special id `'IFL'` for the "I'm Feeling Lucky" station.
- **num_tracks** – the number of tracks to retrieve
- **recently_played_ids** – a list of recently played track ids retrieved from this station. This avoids playing duplicates.

See *get_all_songs()* for the format of a track dictionary.

Podcasts

Mobileclient.**get_all_podcast_series**(*device_id=None*, *incremental=False*, *include_deleted=None*, *updated_after=None*)
Retrieve list of user-subscribed podcast series.

Parameters

- **device_id** – (optional) defaults to `android_id` from login.

  Otherwise, provide a mobile device id as a string. Android device ids are 16 characters, while iOS ids are uuids with 'ios:' prepended.

  If you have already used Google Music on a mobile device, *Mobileclient.get_registered_devices* will provide at least one working id. Omit `'0x'` from the start of the string if present.

  Registered computer ids (a MAC address) will not be accepted and will 403.

  Providing an unregistered mobile device id will register it to your account, subject to Google's device limits. **Registering a device id that you do not own is likely a violation of the TOS.**

- **incremental** – if True, return a generator that yields lists of at most 1000 podcast series as they are retrieved from the server. This can be useful for presenting a loading bar to a user.
- **include_deleted** – ignored. Will be removed in a future release.
- **updated_after** – a datetime.datetime; defaults to unix epoch

Returns a list of podcast series dicts.

Here is an example podcast series dict:

```
{
    'art': [
        {
            'aspectRatio': '1',
            'autogen': False,
            'kind': 'sj#imageRef',
            'url': 'http://lh3.googleusercontent.com/
↪bNoyxoGTwCGkUscMjHsvKe5W80uMOfq...'
        }
    ],
    'author': 'Chris Hardwick',
    'continuationToken': '',
    'description': 'I am Chris Hardwick. I am on TV a lot and have a blog at '
                   'nerdist.com. This podcast is basically just me talking about '
                   'stuff and things with my two nerdy friends Jonah Ray and Matt
↪'
                   'Mira, and usually someone more famous than all of us. '
                   'Occasionally we swear because that is fun. I hope you like '
                   "it, but if you don't I'm sure you will not hesitate to unfurl
↪"
                   "your rage in the 'reviews' section because that's how the "
                   'Internet works.',
    'explicitType': '1',
    'link': 'http://nerdist.com/',
    'seriesId': 'Iliyrhelw74vdqrro77kq2vrdhy',
    'title': 'The Nerdist',
    'totalNumEpisodes': 829,
    'userPreferences': {
        'autoDownload': False,
        'notifyOnNewEpisode': False,
        'subscribed': True
    }
}
```

`Mobileclient.`**`get_all_podcast_episodes`**(*device_id=None*, *incremental=False*, *include_deleted=None*, *updated_after=None*)

Retrieve list of episodes from user-subscribed podcast series.

Parameters

- **`device_id`** – (optional) defaults to `android_id` from login.

  Otherwise, provide a mobile device id as a string. Android device ids are 16 characters, while iOS ids are uuids with 'ios:' prepended.

  If you have already used Google Music on a mobile device, *`Mobileclient.get_registered_devices`* will provide at least one working id. Omit `'0x'` from the start of the string if present.

  Registered computer ids (a MAC address) will not be accepted and will 403.

  Providing an unregistered mobile device id will register it to your account, subject to Google's device limits. **Registering a device id that you do not own is likely a violation of the TOS.**

- **`incremental`** – if True, return a generator that yields lists of at most 1000 podcast

episodes as they are retrieved from the server. This can be useful for presenting a loading bar to a user.

- **include_deleted** – ignored. Will be removed in a future release.
- **updated_after** – a datetime.datetime; defaults to unix epoch

Returns a list of podcast episode dicts.

Here is an example podcast episode dict:

```
{
    'art': [
        {
            'aspectRatio': '1',
            'autogen': False,
            'kind': 'sj#imageRef',
            'url': 'http://lh3.googleusercontent.com/
↪bNoyxoGTwCGkUscMjHsvKe5W80uMOfq...'
        }
    ],
    'deleted': False,
    'description': 'Comedian Bill Burr yelled at Philadelphia, Chris vaguely '
                   'understands hockey, Jonah understands it even less, and Matt '
                   'is weirdly not tired of the running "Matt loves the Dave '
                   'Matthews Band" joke, though I'm sure all of you are.',
    'durationMillis': '4310000',
    'episodeId': 'D6i26frpxu53t2ws3lpbjtpovum',
    'explicitType': '2',
    'fileSize': '69064793',
    'publicationTimestampMillis': '1277791500000',
    'seriesId': 'Iliyrhelw74vdqrro77kq2vrdhy',
    'seriesTitle': 'The Nerdist',
    'title': 'Bill Burr'
}
```

Mobileclient.**add_podcast_series**(*podcast_id*, *notify_on_new_episode=False*)

Subscribe to a podcast series.

Parameters

- **podcast_id** – A podcast series id (hint: they always start with 'I').
- **notify_on_new_episode** – Get device notifications on new episodes.

Returns podcast series id of added podcast series

Mobileclient.**delete_podcast_series**(*podcast_id*)

Unsubscribe to a podcast series.

Parameters **podcast_id** – A podcast series id (hint: they always start with 'I').

Returns podcast series id of removed podcast series

Mobileclient.**edit_podcast_series**(*podcast_id*, *subscribe=True*, *notify_on_new_episode=False*)

Edit a podcast series subscription.

Parameters

- **podcast_id** – A podcast series id (hint: they always start with 'I').
- **subscribe** – Subscribe to podcast.
- **notify_on_new_episode** – Get device notifications on new episodes.

Returns podcast series id of edited podcast series

`Mobileclient.`**`get_podcast_episode_stream_url`**(*podcast_episode_id*, *device_id=None*, *quality=u'hi'*)

Returns a url that will point to an mp3 file.

Parameters

- **`podcast_episde_id`** – a single podcast episode id (hint: they always start with 'D').
- **`device_id`** – (optional) defaults to `android_id` from login.

  Otherwise, provide a mobile device id as a string. Android device ids are 16 characters, while iOS ids are uuids with 'ios:' prepended.

  If you have already used Google Music on a mobile device, *Mobileclient.get_registered_devices* will provide at least one working id. Omit `'0x'` from the start of the string if present.

  Registered computer ids (a MAC address) will not be accepted and will 403.

  Providing an unregistered mobile device id will register it to your account, subject to Google's device limits. **Registering a device id that you do not own is likely a violation of the TOS.**
- **`quality`** – (optional) stream bits per second quality One of three possible values, hi: 320kbps, med: 160kbps, low: 128kbps. The default is hi

When handling the resulting url, keep in mind that:

- you will likely need to handle redirects
- the url expires after a minute
- only one IP can be streaming music at once. This can result in an http 403 with `X-Rejected-Reason: ANOTHER_STREAM_BEING_PLAYED`.

The file will not contain metadata.

Search

Search Google Play for information about artists, albums, tracks, and more.

`Mobileclient.`**`search`**(*query*, *max_results=None*)

Queries Google Music for content.

Parameters

- **`query`** – a string keyword to search with. Capitalization and punctuation are ignored.
- **`max_results`** – Maximum number of items to be retrieved. The maximum accepted value is 100. If set higher, results are limited to 10. A value of `None` allows up to 999 results per type. Default is `None`.

The results are returned in a dictionary with keys: ''album_hits, artist_hits, playlist_hits, podcast_hits,

situation_hits, song_hits, station_hits, video_hits''

containing lists of results of that type.

Free account search is restricted so may not contain hits for all result types.

Here is a sample of results for a search of `'workout'`:

```
{
    'album_hits': [{
        'album': {
            'albumArtRef': 'http://lh5.ggpht.com/DVIg4GiD6msHfgPs_Vu_2eRxCyAoz0fF.
↪..',
            'albumArtist': 'J.Cole',
            'albumId': 'Bfp2tuhynyqppnp6zennhmf6w3y',
            'artist': 'J.Cole',
            'artistId': ['Ajgnxme45wcqqv44vykrleifpji'],
            'description_attribution': {
                'kind': 'sj#attribution',
                'license_title': 'Creative Commons Attribution CC-BY',
                'license_url': 'http://creativecommons.org/licenses/by/4.0/
↪legalcode',
                'source_title': 'Freebase',
                'source_url': ''
            },
            'explicitType': '1',
            'kind': 'sj#album',
            'name': 'Work Out',
            'year': 2011
        },
        'type': '3'
    }],
    'artist_hits': [{
        'artist': {
            'artistArtRef': 'http://lh3.googleusercontent.com/MJe-cDw9uQ-pUagoLlm.
↪..',
            'artistArtRefs': [{
                'aspectRatio': '2',
                'autogen': False,
                'kind': 'sj#imageRef',
                'url': 'http://lh3.googleusercontent.com/MJe-cDw9uQ-pUagoLlmKX3x_
↪K...'
            }],
            'artistId': 'Ajgnxme45wcqqv44vykrleifpji',
            'artist_bio_attribution': {
                'kind': 'sj#attribution',
                'source_title': 'David Jeffries, Rovi'
            },
            'kind': 'sj#artist',
            'name': 'J. Cole'
        },
        'type': '2'
    }],
    'playlist_hits': [{
        'playlist': {
            'albumArtRef': [
                {'url': 'http://lh3.googleusercontent.com/KJsAhrg8Jk_5A4xYLA68LFC.
↪..'}
            ],
            'description': 'Workout Plan ',
            'kind': 'sj#playlist',
            'name': 'Workout',
            'ownerName': 'Ida Sarver',
            'shareToken': 'AMaBXyktyF6Yy_G-8wQy8Rru0tkueIbIFblt2h0BpkvTzHDz-fFj6P.
↪..',
            'type': 'SHARED'
```

```
        },
        'type': '4'
    }],
    'podcast_hits': [{
        'series': {
            'art': [
                {
                    'aspectRatio': '1',
                    'autogen': False,
                    'kind': 'sj#imageRef',
                    'url': 'https://lh3.googleusercontent.com/je4lsaiQCdfcOWoYm3Z_
↪mC...'
                }
            ],
            'author': 'Steve Boyett',
            'continuationToken': '',
            'copyright': 'Music copyright c the respective artists. All other '
                         'material c2006, 2016 by Podrunner, LLC. All rights '
                         'reserved. For personal use only. Unauthorized '
                         'reproduction, sale, rental, exchange, public '
                         'performance, or broadcast of this audio is '
                         'prohibited.',
            'description': 'Nonstop, one-hour, high-energy workout music mixes '
                           "to help you groove while you move. Podrunner's "
                           'fixed-tempo and interval exercise mixes are '
                           'perfect for power walking, jogging, running, '
                           'spinning, elliptical, aerobics, and many other '
                           'tempo-based forms of exercise. An iTunes '
                           'award-winner six years in a row!',
            'explicitType': '2',
            'link': 'http://www.podrunner.com/',
            'seriesId': 'Ilx4ufdua5rdvzplnojtloulo3a',
            'title': 'PODRUNNER: Workout Music',
            'totalNumEpisodes': 0
        },
        'type': '9'
    }],
    'situation_hits': [{
        'situation': {
            'description':
                'Level up and enter beast mode with some loud, aggressive music.',
            'id': 'Nrklpcyfewwrmodvtds5qlfp5ve',
            'imageUrl': 'http://lh3.googleusercontent.com/Cd8WRMaG_pDwjTC_dSPIIuf.
↪..',
            'title': 'Entering Beast Mode',
            'wideImageUrl': 'http://lh3.googleusercontent.com/8A9S-nTb5pfJLcpS8P..
↪.'},
        'type': '7'
    }],
    'song_hits': [{
        'track': {
            'album': 'Work Out',
            'albumArtRef': [{
                'aspectRatio': '1',
                'autogen': False,
                'kind': 'sj#imageRef',
                'url': 'http://lh5.ggpht.com/DVIg4GiD6msHfgPs_Vu_
↪2eRxCyAoz0fFdxj5w...'
```

```
            }],
            'albumArtist': 'J.Cole',
            'albumAvailableForPurchase': True,
            'albumId': 'Bfp2tuhynyqppnp6zennhmf6w3y',
            'artist': 'J Cole',
            'artistId': ['Ajgnxme45wcqqv44vykrleifpji',
↪'Ampniqsqcwxk7btbgh5ycujij5i'],
            'composer': '',
            'discNumber': 1,
            'durationMillis': '234000',
            'estimatedSize': '9368582',
            'explicitType': '1',
            'genre': 'Pop',
            'kind': 'sj#track',
            'nid': 'Tq3nsmzeumhilpegkimjcnbr6aq',
            'primaryVideo': {
                'id': '6PN78PS_QsM',
                'kind': 'sj#video',
                'thumbnails': [{
                    'height': 180,
                    'url': 'https://i.ytimg.com/vi/6PN78PS_QsM/mqdefault.jpg',
                    'width': 320
                }]
            },
            'storeId': 'Tq3nsmzeumhilpegkimjcnbr6aq',
            'title': 'Work Out',
            'trackAvailableForPurchase': True,
            'trackAvailableForSubscription': True,
            'trackNumber': 1,
            'trackType': '7',
            'year': 2011
        },
        'type': '1'
    }],
    'station_hits': [{
        'station': {
            'compositeArtRefs': [{
                    'aspectRatio': '1',
                    'kind': 'sj#imageRef',
                    'url': 'http://lh3.googleusercontent.com/3aD9mFppy6PwjADnjwv_
↪w...'
            }],
            'contentTypes': ['1'],
            'description':
                'These riff-tastic metal tracks are perfect '
                'for getting the blood pumping.',
            'imageUrls': [{
                'aspectRatio': '1',
                'autogen': False,
                'kind': 'sj#imageRef',
                'url': 'http://lh5.ggpht.com/
↪YNGkFdrtk43e8H941fuAHjflrNZ1CJUeqdoys...'
            }],
            'kind': 'sj#radioStation',
            'name': 'Heavy Metal Workout',
            'seed': {
                'curatedStationId': 'Lcwg73w3bd64hsrgarnorif52r',
                'kind': 'sj#radioSeed',
```

```
                'seedType': '9'
            },
            'skipEventHistory': [],
            'stationSeeds': [{
                'curatedStationId': 'Lcwg73w3bd64hsrgarnorif52r',
                'kind': 'sj#radioSeed',
                'seedType': '9'}
            ]},
        'type': '6'
    }],
    'video_hits': [{
        'score': 629.6226806640625,
        'type': '8',
        'youtube_video': {
            'id': '6PN78PS_QsM',
            'kind': 'sj#video',
            'thumbnails': [{
                'height': 180,
                'url': 'https://i.ytimg.com/vi/6PN78PS_QsM/mqdefault.jpg',
                'width': 320
            }],
            'title': 'J. Cole - Work Out'
        }
    }]
}
```

Mobileclient.**get_genres**(*parent_genre_id=None*)

Retrieves information on Google Music genres.

Parameters **parent_genre_id** – (optional) If provided, only child genres will be returned. By default, all root genres are returned. If this id is invalid, an empty list will be returned.

Returns a list of dicts of the form, eg:

```
{
    'name': 'Alternative/Indie',
    'id': 'ALTERNATIVE_INDIE'
    'kind': 'sj#musicGenre',
    'children': [             # this key may not be present
        'ALTERNATIVE_80S',    # these are ids
        'ALT_COUNTRY',
        # ...
        ],
    'images': [
        {
            # these are album covers representative of the genre
            'url': 'http://lh6.ggpht.com/...'
        },
        # ...
    ],
}
```

Note that the id can be used with *create_station()* to seed a radio station.

Mobileclient.**get_album_info**(*album_id*, *include_tracks=True*)

Retrieves details on an album.

Parameters

- **album_id** – an album id (hint: they always start with 'B')

- **include_tracks** – when True, create the `'tracks'` substructure

Returns a dict, eg:

```
{
    'kind': 'sj#album',
    'name': 'Circle',
    'artist': 'Amorphis',
    'albumArtRef': 'http://lh6.ggpht.com/...',
    'tracks': [  # if `include_tracks` is True
    {
        'album': 'Circle',
        'kind': 'sj#track',
        'storeId': 'T5zb7luo2vkroozmj57g2nljdsy',  # can be used as a song id
        'artist': 'Amorphis',
        'albumArtRef': [
        {
            'url': 'http://lh6.ggpht.com/...'
        }],
        'title': 'Shades of Grey',
        'nid': 'T5zb7luo2vkroozmj57g2nljdsy',
        'estimatedSize': '13115591',
        'albumId': 'Bfr2onjv7g7tm4rzosewnnwxxyy',
        'artistId': ['Apoecs6off3y6k4h5nvqqos4b5e'],
        'albumArtist': 'Amorphis',
        'durationMillis': '327000',
        'composer': '',
        'genre': 'Metal',
        'trackNumber': 1,
        'discNumber': 1,
        'trackAvailableForPurchase': True,
        'trackType': '7',
        'albumAvailableForPurchase': True
    }, # ...
    ],
    'albumId': 'Bfr2onjv7g7tm4rzosewnnwxxyy',
    'artistId': ['Apoecs6off3y6k4h5nvqqos4b5e'],
    'albumArtist': 'Amorphis',
    'year': 2013
}
```

Mobileclient.**get_artist_info**(*artist_id*, *include_albums=True*, *max_top_tracks=5*, *max_rel_artist=5*)

Retrieves details on an artist.

Parameters

- **artist_id** – an artist id (hint: they always start with 'A')
- **include_albums** – when True, create the `'albums'` substructure
- **max_top_tracks** – maximum number of top tracks to retrieve
- **max_rel_artist** – maximum number of related artists to retrieve

Returns a dict, eg:

```
{
  'albums':[  # only if include_albums is True
    {
      'albumArtRef':'http://lh6.ggpht.com/...',
```

```
      'albumArtist':'Amorphis',
      'albumId':'Bfr2onjv7g7tm4rzosewnnwxxyy',
      'artist':'Amorphis',
      'artistId':[
        'Apoecs6off3y6k4h5nvqqos4b5e'
      ],
      'kind':'sj#album',
      'name':'Circle',
      'year':2013
    },
  ],
  'artistArtRef':  'http://lh6.ggpht.com/...',
  'artistId':'Apoecs6off3y6k4h5nvqqos4b5e',
  'kind':'sj#artist',
  'name':'Amorphis',
  'related_artists':[  # only if max_rel_artists > 0
    {
      'artistArtRef':      'http://lh5.ggpht.com/...',
      'artistId':'Aheqc7kveljtq7rptd7cy5gvk2q',
      'kind':'sj#artist',
      'name':'Dark Tranquillity'
    }
  ],
  'topTracks':[  # only if max_top_tracks > 0
    {
      'album':'Skyforger',
      'albumArtRef':[
        {
          'url':          'http://lh4.ggpht.com/...'
        }
      ],
      'albumArtist':'Amorphis',
      'albumAvailableForPurchase':True,
      'albumId':'B5nc22xlcmdwi3zn5htkohstg44',
      'artist':'Amorphis',
      'artistId':[
        'Apoecs6off3y6k4h5nvqqos4b5e'
      ],
      'discNumber':1,
      'durationMillis':'253000',
      'estimatedSize':'10137633',
      'kind':'sj#track',
      'nid':'Tn2ugrgkeinrrb2a4ji7khungoy',
      'playCount':1,
      'storeId':'Tn2ugrgkeinrrb2a4ji7khungoy',
      'title':'Silver Bride',
      'trackAvailableForPurchase':True,
      'trackNumber':2,
      'trackType':'7'
    }
  ],
  'total_albums':21
}
```

Mobileclient.**get_podcast_episode_info**(*podcast_episode_id*)

Retrieves information about a podcast episode.

Parameters **podcast_episode_id** – A podcast episode id (hint: they always start with 'D').

Returns a dict, eg:

```
{
    'art': [
        {
            'aspectRatio': '1',
            'autogen': False,
            'kind': 'sj#imageRef',
            'url': 'http://lh3.googleusercontent.com/bNoyxoGTwCGkUscMjHsvKe5...'
        }
    ],
    'description': 'Sarah Jessica Parker (Sex and the City) '
                   'chats with Chris about growing up without '
                   'television, her time on Square Pegs and '
                   'her character in L.A. Story. Sarah Jessica '
                   'then talks about how she felt when she first '
                   'got the part of Carrie on Sex and the '
                   'City, how she dealt with her sudden '
                   'celebrity of being Carrie Bradshaw and they '
                   'come up with a crazy theory about the show! '
                   'They also talk about Sarah Jessica's new '
                   'show Divorce on HBO!',
    'durationMillis': '5101000',
    'episodeId': 'Dcz67vtkhrerzh4hptfqpadt5vm',
    'explicitType': '1',
    'fileSize': '40995252',
    'publicationTimestampMillis': '1475640000000',
    'seriesId': 'Iliyrhelw74vdqrro77kq2vrdhy',
    'seriesTitle': 'The Nerdist',
    'title': 'Sarah Jessica Parker'
}
```

Mobileclient.**get_podcast_series_info**(*podcast_series_id*, *max_episodes=50*)

Retrieves information about a podcast series.

Parameters

- **podcast_series_id** – A podcast series id (hint: they always start with 'I').
- **max_episodes** – Maximum number of episodes to retrieve

Returns a dict, eg:

```
{
    'art': [
        {
            'aspectRatio': '1',
            'autogen': False,
            'kind': 'sj#imageRef',
            'url': 'http://lh3.googleusercontent.com/
↪bNoyxoGTwCGkUscMjHsvKe5W80uMOfq...'
        }
    ],
    'author': 'Chris Hardwick',
    'continuationToken': '',
    'description': 'I am Chris Hardwick. I am on TV a lot and have a blog at '
                   'nerdist.com. This podcast is basically just me talking about '
                   'stuff and things with my two nerdy friends Jonah Ray and Matt
↪'
                   'Mira, and usually someone more famous than all of us. '
```

```
                    'Occasionally we swear because that is fun. I hope you like '
                    "it, but if you don't I'm sure you will not hesitate to unfurl
↪"
                    "your rage in the 'reviews' section because that's how the "
                    'Internet works.',
    'episodes': [
        {
            'art': [
                {
                 'aspectRatio': '1',
                 'autogen': False,
                 'kind': 'sj#imageRef',
                 'url': 'http://lh3.googleusercontent.com/
↪bNoyxoGTwCGkUscMjHsvKe5...'
                }
            ],
            'description': 'Sarah Jessica Parker (Sex and the City) '
                           'chats with Chris about growing up without '
                           'television, her time on Square Pegs and '
                           'her character in L.A. Story. Sarah Jessica '
                           'then talks about how she felt when she first '
                           'got the part of Carrie on Sex and the '
                           'City, how she dealt with her sudden '
                           'celebrity of being Carrie Bradshaw and they '
                           'come up with a crazy theory about the show! '
                           'They also talk about Sarah Jessica's new '
                           'show Divorce on HBO!',
            'durationMillis': '5101000',
            'episodeId': 'Dcz67vtkhrerzh4hptfqpadt5vm',
            'explicitType': '1',
            'fileSize': '40995252',
            'publicationTimestampMillis': '1475640000000',
            'seriesId': 'Iliyrhelw74vdqrro77kq2vrdhy',
            'seriesTitle': 'The Nerdist',
            'title': 'Sarah Jessica Parker'
        },
    ]
    'explicitType': '1',
    'link': 'http://nerdist.com/',
    'seriesId': 'Iliyrhelw74vdqrro77kq2vrdhy',
    'title': 'The Nerdist',
    'totalNumEpisodes': 829,
    'userPreferences': {
        'autoDownload': False,
        'notifyOnNewEpisode': False,
        'subscribed': True
    }
}
```

Mobileclient.**get_track_info**(*store_track_id*)

Retrieves information about a store track.

Parameters **store_track_id** – a store track id (hint: they always start with 'T')

Returns a dict, eg:

```
{
    'album': 'Best Of',
    'kind': 'sj#track',
```

```
    'storeId': 'Te2qokfjmhqxw4bnkswbfphzs4m',
    'artist': 'Amorphis',
    'albumArtRef': [
    {
        'url': 'http://lh5.ggpht.com/...'
    }],
    'title': 'Hopeless Days',
    'nid': 'Te2qokfjmhqxw4bnkswbfphzs4m',
    'estimatedSize': '12325643',
    'albumId': 'Bsbjjc24a5xutbutvbvg3h4y2k4',
    'artistId': ['Apoecs6off3y6k4h5nvqqos4b5e'],
    'albumArtist': 'Amorphis',
    'durationMillis': '308000',
    'composer': '',
    'genre': 'Metal',
    'trackNumber': 2,
    'discNumber': 1,
    'trackAvailableForPurchase': True,
    'trackType': '7',
    'albumAvailableForPurchase': True
}
```

Mobileclient.**get_station_info**(*station_id*, *num_tracks=25*)

Retrieves information about a station.

Parameters

- **station_id** – a station id
- **include_tracks** – when True, create the `'tracks'` substructure
- **num_tracks** – maximum number of tracks to return

Returns a dict, eg:

```
{
    'kind':'sj#radioStation',
    'byline':'By Google Play Music',
    'name':'Neo Soul',
    'compositeArtRefs':[
        {
            'url':'http://lh3.googleusercontent.com/Aa-
↪WBVTbKegp6McwqeHlj6KX5EYHKOBag74uwNl4xIHSv1g7Mi-NkMzwig',
            'kind':'sj#imageRef',
            'aspectRatio':'2'
        }               ],
    'deleted':False,
    'enforcementResult':{
        'sessionInvalidated':False
    },
    'lastModifiedTimestamp':'1497410517701000',
    'recentTimestamp':'1497410516945000',
    'clientId':'9e66e89e-50b0-11e7-aaa3-bc5ff4545c4b',
    'sessionToken':'AFTSR9PtB_PbyqZ3jsnl-
↪PFma4upK1MEtlhVnIlxRNynGalctoJF4TpgzaxymOnk0Gv5DQG7gb_W3eLamPU_
↪Mg1cWylhrowQi1EFMBKWHeDDYWzpU1cEOF-D3c_gnwsBRHIuOetph2veY2Fd-dKVzjOkN6mtidE-
↪XPR2VnpR9PG83wRLVRtJq5593-Vvbu6wjCHD9f23ohxg-
↪ki0tyD3fjFW1463zy63YzN5Aa2SpbvOskEWhwhS3u9ASgEoX08lePE-
↪ZZAq1XtmVvLa8DnDMVb7i95Qhp0dM2it1uruKHH85u7tMYnttbAW4022d0rqrp3ULDKOYMvIIouXH44-
↪bkbKLuVIADiqeNavwTVzcoJxWo4mMKjCaxM=',
```

```
    'tracks':[
        {
            'albumArtRef':[
                {
                    'url':'http://lh5.ggpht.com/vWRj9DkKZ7cFj-
↪qXoGoBGsv7ngUWdtGNl1SSOdzj2efDwdAs3F0kJ3Xq6zLxKjgv1v3ive5S',
                    'kind':'sj#imageRef',
                    'aspectRatio':'1',
                    'autogen':False
                }
            ],
            'artistId':[
                'Atmjrctnubes5zhftrey2xjkzl'
            ],
            'composer':'',
            'year':1996,
            'trackAvailableForSubscription':True,
            'trackType':'7',
            'album':u"Maxwell's Urban Hang Suite",
            'title':u"Ascension (Don't Ever Wonder)",
            'albumArtist':'Maxwell',
            'trackNumber':4,
            'discNumber':1,
            'albumAvailableForPurchase':False,
            'explicitType':'2',
            'trackAvailableForPurchase':True,
            'storeId':'T6utayayrlyfmpovgj4ulacpat',
            'nid':'T6utayayrlyfmpovgj4ulacpat',
            'estimatedSize':'13848059',
            'albumId':'Bpwzztxynfjwtnrtgiugem3b56e',
            'genre':'Neo-Soul',
            'kind':'sj#track',
            'primaryVideo':{
                'kind':'sj#video',
                'id':'D7rm9t5S4uE',
                'thumbnails':[
                    {
                        'url':'https://i.ytimg.com/vi/D7rm9t5S4uE/mqdefault.jpg',
                        'width':320,
                        'height':180
                    }
                ]
            },
            'artist':'Maxwell',
            'wentryid':'ec9428eb-2676-4e92-901d-2de9a72fe581',
            'durationMillis':'346000'
        }
    ],
    'seed':{
        'kind':'sj#radioSeed',
        'curatedStationId':'L3lu7bpcqtd3e7pa7w37rf7gdu',
        'seedType':'9'
    },
    'skipEventHistory':[
    ],
    'inLibrary':False,
    'imageUrls':[
        {
```

```
            'url':'http://lh3.googleusercontent.com/
↪iceDDsQjQ683AD4w21WWlekg115Ixy_kMTivkFJTjo3w7vuW4-SSs3F3KQOaR8qoI-QYVuOQoA',
            'kind':'sj#imageRef',
            'aspectRatio':'1',
            'autogen':False
        }
    ],
    'id':'1a9ec96c-6c98-3c43-b123-9e2743203f5d'
}
```

Misc

Mobileclient.**get_browse_podcast_hierarchy**()

Retrieve the hierarchy of podcast browse genres.

Returns a list of podcast genres and subgenres:

```
{
    "groups": [
        {
            "id": "JZCpodcasttopchart",
            "displayName": "Top Charts",
            "subgroups": [
                {
                 "id": "JZCpodcasttopchartall",
                 "displayName": "All categories"
                },
                {
                 "id": "JZCpodcasttopchartarts",
                 "displayName": "Arts"
                },
                {
                 "id": "JZCpodcasttopchartbusiness",
                 "displayName": "Business"
                },
                {
                 "id": "JZCpodcasttopchartcomedy",
                 "displayName": "Comedy"
                },
                {
                 "id": "JZCpodcasttopcharteducation",
                 "displayName": "Education"
                },
                {
                 "id": "JZCpodcasttopchartgames",
                 "displayName": "Games & hobbies"
                },
                {
                 "id": "JZCpodcasttopchartgovernment",
                 "displayName": "Government & organizations"
                },
                {
                 "id": "JZCpodcasttopcharthealth",
                 "displayName": "Health"
                },
                {
```

```
                    "id": "JZCpodcasttopchartkids",
                    "displayName": "Kids & families"
                },
                {
                    "id": "JZCpodcasttopchartmusic",
                    "displayName": "Music"
                },
                {
                    "id": "JZCpodcasttopchartnews",
                    "displayName": "News & politics"
                },
                {
                    "id": "JZCpodcasttopchartreligion",
                    "displayName": "Religion & spirituality"
                },
                {
                    "id": "JZCpodcasttopchartscience",
                    "displayName": "Science & medicine"
                },
                {
                    "id": "JZCpodcasttopchartsociety",
                    "displayName": "Society & culture"
                },
                {
                    "id": "JZCpodcasttopchartsports",
                    "displayName": "Sports & recreation"
                },
                {
                    "id": "JZCpodcasttopcharttechnology",
                    "displayName": "Technology"
                },
                {
                    "id": "JZCpodcasttopcharttv",
                    "displayName": "TV & film"
                }
            ]
        }
    ]
}
```


`Mobileclient.`**`get_browse_podcast_series`**`(`*genre_id=u'JZCpodcasttopchartall'*`)`

Retrieve podcast series from browse podcasts by genre.

Parameters **`genre_id`** – A podcast genre id as returned by `get_podcast_browse_hierarchy()`. Defaults to Top Chart 'All categories'.

Returns a list of podcast series dicts.

Here is an example podcast series dict:


```
{
    'art': [
        {
            'aspectRatio': '1',
            'autogen': False,
            'kind': 'sj#imageRef',
            'url': 'http://lh3.googleusercontent.com/liR-
↪Pm7EhB58wrAa4uo9Y33LcJJ8keU...'
        }
```

```
    ],
    'author': 'NBC Sports Radio',
    'continuationToken': '',
    'description': 'Mike Florio talks about the biggest NFL topics with the '
                   'people who are most passionate about the game: League execs, '
                   'players, coaches and the journalists who cover pro football.',
    'explicitType': '2',
    'link': 'https://audioboom.com/channel/pro-football-talk-live-with-mike-florio
↪',
    'seriesId': 'I3iad5heqorm3nck6yp7giruc5i',
    'title': 'Pro Football Talk Live with Mike Florio',
    'totalNumEpisodes': 0
}
```

Mobileclient.**get_listen_now_items**()

Returns a list of dictionaries of Listen Now albums and stations.

See *get_listen_now_situations()* for Listen Now situations.

Here is an example Listen Now album:

```
{
  'album': {
    'artist_metajam_id': 'A2mfgoustq7iqjdbvlenw7pnap4',
    'artist_name': 'Justin Bieber',
    'artist_profile_image': {
      'url': 'http://lh3.googleusercontent.com/XgktDR74DWE9xD...'',
    },
    'description': 'Purpose is the fourth studio album by Canadian...',
    'description_attribution': {
      'kind': 'sj#attribution',
      'license_title': 'Creative Commons Attribution CC-BY-SA 4.0',
      'license_url': 'http://creativecommons.org/licenses/by-sa/4.0/legalcode',
      'source_title': 'Wikipedia',
      'source_url': 'http://en.wikipedia.org/wiki/Purpose_(Justin_Bieber_album)',
    },
    'id': {
      'artist': 'Justin Bieber',
      'metajamCompactKey': 'Bqpez5cimsze2fh6w7j2rcf55xa',
      'title': 'Purpose (Deluxe)',
    },
    'title': 'Purpose (Deluxe)'
    'images': [
      {
        'kind': 'sj#imageRef',
        'url': 'http://lh3.googleusercontent.com/m66cbl4Jl3VNz...',
      },
    ],
  }
  'kind': 'sj#listennowitem',
  'suggestion_reason': '9',
  'suggestion_text': 'Popular album on Google Play Music',
  'type': '1'
}
```

Here is an example Listen Now station:

```
{
  'radio_station': {
    'id': {
      'seeds': [
        {
          'artistId': 'Ax6ociylvowozcz2iepfqsar54i',
          'kind': 'sj#radioSeed',
          'metadataSeed': {
            'artist': {
              'artistArtRef': 'http://lh3.googleusercontent.com/x9qukAx...',
              'artistArtRefs': [
                {
                  'aspectRatio': '2',
                  'autogen': False,
                  'kind': 'sj#imageRef',
                  'url': 'http://lh3.googleusercontent.com/x9qukAx...',
                },
              ],
              'artistId': 'Ax6ociylvowozcz2iepfqsar54i',
              'artist_bio_attribution': {
              'kind': 'sj#attribution',
              'source_title': 'artist representative',
              },
              'kind': 'sj#artist',
              'name': 'Drake',
            },
            'kind': 'sj#radioSeedMetadata',
          },
         'seedType': '3',
        },
      ]
    },
    'title': 'Drake',
  },
  'compositeArtRefs': [
    {
      'aspectRatio': '2',
      'kind': 'sj#imageRef',
      'url': 'http://lh3.googleusercontent.com/rE39ky1yZN...',
    },
    {
      'aspectRatio': '1',
      'kind': 'sj#imageRef',
      'url': 'http://lh3.googleusercontent.com/Pcwg_HngBr...',
    },
  ],
  'images': [
    {
      'aspectRatio': '2',
      'autogen': False,
      'kind': 'sj#imageRef',
      'url': 'http://lh3.googleusercontent.com/x9qukAx_TMam...',
    },
  ],
  'suggestion_reason': '9',
  'suggestion_text': 'Popular artist on Google Play Music',
  'type': '3'
}
```

Mobileclient.**get_listen_now_situations**()

Returns a list of dictionaries that each represent a Listen Now situation.

See *get_listen_now_items()* for Listen Now albums and stations.

A situation contains a list of related stations or other situations.

Here is an example situation:


```
{
    'description': 'Select a station of today's most popular songs.',
    'id': 'Ntiiwllegkw73p27o236mfsj674',
    'imageUrl': 'http://lh3.googleusercontent.com/egm4NgIK-Cmh84GjVgH...',
    'stations': [
        {
            'compositeArtRefs': [
                {
                    'aspectRatio': '2',
                    'kind': 'sj#imageRef',
                    'url': 'http://lh3.googleusercontent.com/ffDI377y...',
                },
            ],
            'contentTypes': ['1'],
            'description': "This playlist features today's biggest pop songs...",
            'imageUrls': [
                {
                    'aspectRatio': '1',
                    'autogen': False,
                    'kind': 'sj#imageRef',
                    'url': 'http://lh3.googleusercontent.com/B4iKX23Z...',
                },
            ],
            'kind': 'sj#radioStation',
            'name': "Today's Pop Hits",
            'seed': {
                'curatedStationId': 'Lgen6kdn43tz5b3edimqd5e4ckq',
                'kind': 'sj#radioSeed',
                'seedType': '9',
            },
            'skipEventHistory': [],
            'stationSeeds': [
                {
                    'curatedStationId': 'Lgen6kdn43tz5b3edimqd5e4ckq',
                    'kind': 'sj#radioSeed',
                    'seedType': '9',
                },
            ],
        }
    ],
    'title': "Today's Biggest Hits",
    'wideImageUrl': 'http://lh3.googleusercontent.com/13W-bm3sNmSfOjUkEqY...'
}
```


Musicmanager Interface

class gmusicapi.clients.**Musicmanager**(*debug_logging=True*, *validate=True*, *verify_ssl=True*)

Allows uploading by posing as Google's Music Manager.

Musicmanager uses OAuth, so a plaintext email and password are not required when logging in.

For most authors and users of gmusicapi scripts, *perform_oauth()* should be run once per machine to store credentials to disk. Future calls to *login()* can use use the stored credentials by default.

Some authors may want more control over the OAuth flow. In this case, credentials can be directly provided to *login()*.

Setup and login

static Musicmanager.**perform_oauth**(*storage_filepath=u'/home/docs/.local/share/gmusicapi/oauth.cred'*, *open_browser=False*)

Provides a series of prompts for a user to follow to authenticate. Returns `oauth2client.client.OAuth2Credentials` when successful.

In most cases, this should only be run once per machine to store credentials to disk, then never be needed again.

If the user refuses to give access, `oauth2client.client.FlowExchangeError` is raised.

Parameters

- **storage_filepath** – a filepath to write the credentials to, or `None` to not write the credentials to disk (which is not recommended).

  Appdirs `user_data_dir` is used by default. Users can run:

  ```
  import gmusicapi.clients
  print gmusicapi.clients.OAUTH_FILEPATH
  ```

  to see the exact location on their system.

- **open_browser** – if True, attempt to open the auth url in the system default web browser. The url will be printed regardless of this param's setting.

This flow is intentionally very simple. For complete control over the OAuth flow, pass an `oauth2client.client.OAuth2Credentials` to *login()* instead.

Musicmanager.**__init__**(*debug_logging=True*, *validate=True*, *verify_ssl=True*)

Parameters

- **debug_logging** – each Client has a `logger` member. The logger is named `gmusicapi.<client class><client number>` and will propogate to the `gmusicapi` root logger.

  If this param is `True`, handlers will be configured to send this client's debug log output to disk, with warnings and above printed to stderr. Appdirs `user_log_dir` is used by default. Users can run:

  ```
  from gmusicapi.utils import utils
  print utils.log_filepath
  ```

  to see the exact location on their system.

  If `False`, no handlers will be configured; users must create their own handlers.

  Completely ignoring logging is dangerous and not recommended. The Google Music protocol can change at any time; if something were to go wrong, the logs would be necessary for recovery.

- **validate** – if False, do not validate server responses against known schemas. This helps to catch protocol changes, but requires significant cpu work.

  This arg is stored as `self.validate` and can be safely modified at runtime.

- **verify_ssl** – if False, exceptions will not be raised if there are problems verifying SSL certificates. Be wary of using this option; it's almost always better to fix the machine's SSL configuration than to ignore errors.

`Musicmanager.`**`login`**(*oauth_credentials=u'/home/docs/.local/share/gmusicapi/oauth.cred'*, *uploader_id=None*, *uploader_name=None*)

Authenticates the Music Manager using OAuth. Returns `True` on success, `False` on failure.

Unlike the *Webclient*, OAuth allows authentication without providing plaintext credentials to the application.

In most cases, the default parameters should be acceptable. Users on virtual machines will want to provide *uploader_id*.

Parameters

- **oauth_credentials** – `oauth2client.client.OAuth2Credentials` or the path to a `oauth2client.file.Storage` file. By default, the same default path used by *perform_oauth()* is used.

  Endusers will likely call *perform_oauth()* once to write credentials to disk and then ignore this parameter.

  This param is mostly intended to allow flexibility for developers of a 3rd party service who intend to perform their own OAuth flow (eg on their website).

- **uploader_id** – a unique id as a MAC address, eg `'00:11:22:33:AA:BB'`. This should only be provided in cases where the default (host MAC address incremented by 1) will not work.

  Upload behavior is undefined if a Music Manager uses the same id, especially when reporting bad matches.

  `ValueError` will be raised if this is provided but not in the proper form.

  `OSError` will be raised if this is not provided and a real MAC could not be determined (most common when running on a VPS).

  If provided, use the same id on all future runs for this machine, because of the upload device limit explained below.

- **uploader_name** – human-readable non-unique id; default is `"<hostname> (gmusicapi-{version})"`.

  This doesn't appear to be a part of authentication at all. Registering with (id, name = X, Y) and logging in with (id, name = X, Z) works, and does not change the server-stored uploader_name.

There are hard limits on how many upload devices can be registered; refer to Google's docs. There have been limits on deauthorizing devices in the past, so it's smart not to register more devices than necessary.

`Musicmanager.`**`logout`**(*revoke_oauth=False*)

Forgets local authentication in this Client instance.

Parameters **revoke_oauth** – if True, oauth credentials will be permanently revoked. If credentials came from a file, it will be deleted.

Returns `True` on success.

Uploading Songs

Musicmanager.**upload**(*filepaths*, *enable_matching=False*, *enable_transcoding=True*, *transcode_quality=u'320k'*)

Uploads the given filepaths.

All non-mp3 files will be transcoded before being uploaded. This is a limitation of Google's backend.

An available installation of ffmpeg or avconv is required in most cases: see the installation page for details.

Returns a 3-tuple `(uploaded, matched, not_uploaded)` of dictionaries, eg:

```
(
    {'<filepath>': '<new server id>'},               # uploaded
    {'<filepath>': '<new server id>'},               # matched
    {'<filepath>': '<reason, eg ALREADY_EXISTS>'}    # not uploaded
)
```

Parameters

- **filepaths** – a list of filepaths, or a single filepath.
- **enable_matching** – if `True`, attempt to use scan and match to avoid uploading every song. This requires ffmpeg or avconv. **WARNING**: currently, mismatched songs can *not* be fixed with the 'Fix Incorrect Match' button nor *report_incorrect_match*. They would have to be deleted and reuploaded with matching disabled (or with the Music Manager). Fixing matches from gmusicapi may be supported in a future release; see issue #89.
- **enable_transcoding** – if `False`, non-MP3 files that aren't matched using scan and match will not be uploaded.
- **transcode_quality** – if int, pass to ffmpeg/avconv `-q:a` for libmp3lame (lower-better int,). If string, pass to ffmpeg/avconv `-b:a` (eg `'128k'` for an average bitrate of 128k). The default is 320kbps cbr (the highest possible quality).

All Google-supported filetypes are supported; see Google's documentation.

If `PERMANENT_ERROR` is given as a not_uploaded reason, attempts to reupload will never succeed. The file will need to be changed before the server will reconsider it; the easiest way is to change metadata tags (it's not important that the tag be uploaded, just that the contents of the file change somehow).

Downloading Songs

Musicmanager.**get_uploaded_songs**(*incremental=False*)

Returns a list of dictionaries, each with the following keys: `('id', 'title', 'album', 'album_artist', 'artist', 'track_number', 'track_size', 'disc_number', 'total_disc_count')`.

All Access tracks that were added to the library will not be included, only tracks uploaded/matched by the user.

Parameters **incremental** – if True, return a generator that yields lists of at most 1000 dictionaries as they are retrieved from the server. This can be useful for presenting a loading bar to a user.

Musicmanager.**get_purchased_songs**(*incremental=False*)

Returns a list of dictionaries, each with the following keys: `('id', 'title', 'album', 'album_artist', 'artist', 'track_number', 'track_size', 'disc_number', 'total_disc_count')`.

Parameters **incremental** – if True, return a generator that yields lists of at most 1000 dictionaries as they are retrieved from the server. This can be useful for presenting a loading bar to a user.

Musicmanager.**download_song**(*song_id*)

Download an uploaded or purchased song from your library.

Subscription tracks can't be downloaded with this method.

Returns a tuple `(u'suggested_filename', 'audio_bytestring')`. The filename will be what the Music Manager would save the file as, presented as a unicode string with the proper file extension. You don't have to use it if you don't want.

Parameters **song_id** – a single uploaded or purchased song id.

To write the song to disk, use something like:

```
filename, audio = mm.download_song(an_id)

# if open() throws a UnicodeEncodeError, either use
#   filename.encode('utf-8')
# or change your default encoding to something sane =)
with open(filename, 'wb') as f:
    f.write(audio)
```

Unlike with *Webclient.get_song_download_info*, there is no download limit when using this interface.

Also unlike the Webclient, downloading a track requires authentication. Returning a url does not suffice, since retrieving a track without auth will produce an http 500.

Misc

..automethod:: Musicmanager.get_quota

2.2 Support for Other Languages

Here are the ports I'm currently aware of:

- C++: dvirtz and Greg Wicks <https://github.com/gwicks/gmusicapi-curl>'__
- C#: ffleischer and Taylor Finnell
- Go: Lari Rasku
- Java: Jens Villadsen and Nick Martin
- Javascript: Lari Rasku. There's also my Google Music Turntable uploader; it's not a port, but may be useful as an example.
- Node: Jamon Terrell
- Objective-C: Gregory Wicks
- PHP: raydanhk
- Ruby: Loic Nageleisen

They're in various states of completion and maintenance because, well, building a port is tough.

Alternatively, consider using GMusicProxy or copying its approach.

2.2.1 Building a Port

Get in touch if you're working on a port. I'm happy to answer questions and point you to relevant bits of code.

Generally, though, the protocol package is what you'll want to look at. It contains all of the call schemas in a psuedo-dsl that's explained here.

2.3 Contributing to gmusicapi

The easiest way to start contributing is to help triage issues.

2.3.1 Development

Please make pull requests to the `develop` branch. `master` is currently used to hold the code of the most recent release.

Building the docs locally is straightforward. First, make sure Sphinx is installed: `$ pip install sphinx`.

Next, simply do `$ cd gmusicapi/docs` followed by `$ make html`.

If there weren't any problems, the docs are now in `build/html`. You can serve them up locally with `$ python -m SimpleHTTPServer`, then view them in your web browser at `http://127.0.0.1:8000/build/html/`.

2.4 Getting started

The *usage section* has installation instructions and some simple examples.

2.5 Api and data reference

The reference has details for all classes and functions, as well as information on the Google Music data you'll encounter:

2.5.1 Protocol Details

The following definitions represent known endpoints relating to Google Music. They are organized by the client that uses them.

The names of these classes are semantic, and may not match their actual endpoint.

Most of the time, endusers will want to use one of the *Client Interfaces*. However, any of the definitions listed here can be called by using the `_make_call` member of a Client and providing the parameters needed by `dynamic_*` functions.

It's tough to generate the exact schema of every call in a readable fashion, so this information is left out. If you need exact specifications, look at the code - or submit a pull request to generate the docs =)

Android Client

Calls made by the mobile client.

class gmusicapi.protocol.mobileclient.**BatchMutatePlaylistEntries**

static build_plentry_adds(*playlist_id*, *song_ids*)

Parameters

- **playlist_id** –
- **song_ids** –

static build_plentry_deletes(*entry_ids*)

Parameters **entry_ids** –

static build_plentry_reorder(*plentry*, *preceding_cid*, *following_cid*)

Parameters

- **plentry** – plentry that is moving
- **preceding_cid** – clientid of entry that will be before the moved entry
- **following_cid** – "" that will be after the moved entry

static_method = u'POST'

static_url = u'https://mclients.googleapis.com/sj/v2.5/plentriesbatch'

class gmusicapi.protocol.mobileclient.**BatchMutatePlaylists**

static build_playlist_adds(*pl_descriptions*)

Parameters **pl_descriptions** – [{'name': '', 'description': '','public': ''}]

static build_playlist_deletes(*playlist_ids*)

Parameters **playlist_ids** –

static build_playlist_updates(*pl_updates*)

Parameters **pl_updates** – [{'id': '', name': '', 'description': '', 'public': ''}]

static_method = u'POST'

static_url = u'https://mclients.googleapis.com/sj/v2.5/playlistbatch'

class gmusicapi.protocol.mobileclient.**BatchMutatePodcastSeries**

static build_podcast_updates(*updates*)

Parameters **updates** –

[

{'seriesId': '', 'subscribed': '', 'userPreferences': { 'notifyOnNewEpisode': '', 'subscrubed': ''}}...

]

static_method = u'POST'

static_url = u'https://mclients.googleapis.com/sj/v2.5/podcastseries/batchmutate'

class gmusicapi.protocol.mobileclient.**BatchMutateStations**

static build_add(*name*, *seed*, *include_tracks*, *num_tracks*, *recent_datetime=None*)

Parameters

- **name** – the title
- **seed** – a dict {'itemId': id, 'seedType': int}
- **include_tracks** – if True, return *num_tracks* tracks in the response
- **num_tracks** –
- **recent_datetime** – purpose unknown. defaults to now.

static build_deletes(*station_ids*)

Parameters **station_ids** –

static_method = u'POST'

static_url = u'https://mclients.googleapis.com/sj/v2.5/radio/editstation'

class gmusicapi.protocol.mobileclient.**BatchMutateTracks**

static build_track_add(*store_track_info*)

Parameters **store_track_info** – sj_track

static build_track_deletes(*track_ids*)

Parameters **track_ids** –

static_method = u'POST'

static_url = u'https://mclients.googleapis.com/sj/v2.5/trackbatch'

class gmusicapi.protocol.mobileclient.**Config**

static_method = u'GET'

static_url = u'https://mclients.googleapis.com/sj/v2.5/config'

class gmusicapi.protocol.mobileclient.**DeauthDevice**

Deauthorize a device from devicemanagementinfo.

static dynamic_params(*device_id*)

static_method = u'DELETE'

static_url = u'https://mclients.googleapis.com/sj/v2.5/devicemanagementinfo'

class gmusicapi.protocol.mobileclient.**GetAlbum**

static dynamic_params(*album_id*, *tracks*)

static_method = u'GET'

static_params = {u'alt': u'json'}

static_url = u'https://mclients.googleapis.com/sj/v2.5/fetchalbum'

class gmusicapi.protocol.mobileclient.**GetArtist**

static **dynamic_params**(*artist_id*, *include_albums*, *num_top_tracks*, *num_rel_artist*)

Parameters

- **include_albums** – bool
- **num_top_tracks** – int
- **num_rel_artist** – int

static_method = u'GET'

static_params = {u'alt': u'json'}

static_url = u'https://mclients.googleapis.com/sj/v2.5/fetchartist'

class gmusicapi.protocol.mobileclient.**GetBrowsePodcastHierarchy**

static_method = u'GET'

static_params = {u'alt': u'json'}

static_url = u'https://mclients.googleapis.com/sj/v2.5/podcast/browsehierarchy'

class gmusicapi.protocol.mobileclient.**GetDeviceManagementInfo**

Get registered device information.

static_method = u'GET'

static_params = {u'alt': u'json'}

static_url = u'https://mclients.googleapis.com/sj/v2.5/devicemanagementinfo'

class gmusicapi.protocol.mobileclient.**GetGenres**

static **dynamic_params**(*parent_genre_id*)

static_method = u'GET'

static_params = {u'alt': u'json'}

static_url = u'https://mclients.googleapis.com/sj/v2.5/explore/genres'

class gmusicapi.protocol.mobileclient.**GetPodcastEpisode**

static **dynamic_params**(*podcast_episode_id*)

static_headers = {u'Content-Type': u'application/json'}

static_method = u'GET'

static_params = {u'alt': u'json'}

static_url = u'https://mclients.googleapis.com/sj/v2.5/podcast/fetchepisode'

class gmusicapi.protocol.mobileclient.**GetPodcastEpisodeStreamUrl**

static_method = u'GET'

static_url = u'https://mclients.googleapis.com/music/fplay'

class gmusicapi.protocol.mobileclient.**GetPodcastSeries**

static **dynamic_params**(*podcast_series_id*, *num_episodes*)

static_headers = {u'Content-Type': u'application/json'}

static_method = u'GET'

static_params = {u'alt': u'json'}

static_url = u'https://mclients.googleapis.com/sj/v2.5/podcast/fetchseries'

class gmusicapi.protocol.mobileclient.**GetStationTrackStreamUrl**

static dynamic_headers(*item_id*, *wentry_id*, *session_token*, *quality*)

classmethod dynamic_params(*song_id*, *wentry_id*, *session_token*, *quality*)

static parse_response(*response*)

static_method = u'GET'

static_url = u'https://mclients.googleapis.com/music/wplay'

class gmusicapi.protocol.mobileclient.**GetStoreTrack**

static dynamic_params(*track_id*)

static_headers = {u'Content-Type': u'application/json'}

static_method = u'GET'

static_params = {u'alt': u'json'}

static_url = u'https://mclients.googleapis.com/sj/v2.5/fetchtrack'

class gmusicapi.protocol.mobileclient.**GetStreamUrl**

static_method = u'GET'

static_url = u'https://mclients.googleapis.com/music/mplay'

class gmusicapi.protocol.mobileclient.**IncrementPlayCount**

static dynamic_data(*sid*, *plays*, *playtime*)

static_headers = {u'Content-Type': u'application/json'}

static_method = u'POST'

static_params = {u'alt': u'json'}

static_url = u'https://mclients.googleapis.com/sj/v2.5/trackstats'

class gmusicapi.protocol.mobileclient.**ListBrowsePodcastSeries**

classmethod dynamic_params(*id=None*)

static_method = u'GET'

static_params = {u'alt': u'json'}

static_url = u'https://mclients.googleapis.com/sj/v2.5/podcast/browse'

class gmusicapi.protocol.mobileclient.**ListListenNowItems**

static_method = u'GET'

static_params = {u'alt': u'json'}

static_url = u'https://mclients.googleapis.com/sj/v2.5/listennow/getlistennowitems'

class gmusicapi.protocol.mobileclient.**ListListenNowSituations**

classmethod dynamic_data()

static_headers = {u'Content-Type': u'application/json'}

static_method = u'POST'

static_params = {u'alt': u'json'}

static_url = u'https://mclients.googleapis.com/sj/v2.5/listennow/situations'

class gmusicapi.protocol.mobileclient.**ListPlaylistEntries**

static_method = u'POST'

static_url = u'https://mclients.googleapis.com/sj/v2.5/plentryfeed'

class gmusicapi.protocol.mobileclient.**ListPlaylists**

static_method = u'POST'

static_url = u'https://mclients.googleapis.com/sj/v2.5/playlistfeed'

class gmusicapi.protocol.mobileclient.**ListPodcastEpisodes**

classmethod dynamic_data(*device_id=None*, *updated_after=None*, *start_token=None*, *max_results=None*)

static dynamic_headers(*device_id*, *updated_after=None*, *start_token=None*, *max_results=None*)

classmethod dynamic_params(*device_id=None*, *updated_after=None*, *start_token=None*, *max_results=None*)

static_method = u'GET'

static_url = u'https://mclients.googleapis.com/sj/v2.5/podcastepisode'

class gmusicapi.protocol.mobileclient.**ListPodcastSeries**

classmethod dynamic_data(*device_id=None*, *updated_after=None*, *start_token=None*, *max_results=None*)

static dynamic_headers(*device_id*, *updated_after=None*, *start_token=None*, *max_results=None*)

classmethod dynamic_params(*device_id=None*, *updated_after=None*, *start_token=None*, *max_results=None*)

static_method = u'GET'

static_url = u'https://mclients.googleapis.com/sj/v2.5/podcastseries'

class gmusicapi.protocol.mobileclient.**ListPromotedTracks**

static_method = u'POST'

static_url = u'https://mclients.googleapis.com/sj/v2.5/ephemeral/top'

class gmusicapi.protocol.mobileclient.**ListSharedPlaylistEntries**

classmethod dynamic_data(*share_token*, *updated_after=None*, *start_token=None*, *max_results=None*)

Parameters

- **share_token** – from a shared playlist
- **updated_after** – ignored
- **start_token** – nextPageToken from a previous response
- **max_results** – a positive int; if not provided, server defaults to 1000

classmethod dynamic_params(*share_token*, *updated_after=None*, *start_token=None*, *max_results=None*)

classmethod parse_response(*response*)

static_method = u'POST'

static_url = u'https://mclients.googleapis.com/sj/v2.5/plentries/shared'

class gmusicapi.protocol.mobileclient.**ListStationTracks**

static dynamic_data(*station_id*, *num_entries*, *recently_played*)

Parameters

- **station_id** –
- **num_entries** – maximum number of tracks to return
- **recently_played** – a list of...song ids? never seen an example

static_headers = {u'Content-Type': u'application/json'}

static_method = u'POST'

static_params = {u'alt': u'json', u'include-tracks': u'true'}

static_url = u'https://mclients.googleapis.com/sj/v2.5/radio/stationfeed'

class gmusicapi.protocol.mobileclient.**ListStations**

static_method = u'POST'

static_url = u'https://mclients.googleapis.com/sj/v2.5/radio/station'

class gmusicapi.protocol.mobileclient.**ListTracks**

static_method = u'POST'

static_url = u'https://mclients.googleapis.com/sj/v2.5/trackfeed'

class gmusicapi.protocol.mobileclient.**McBatchMutateCall**

Abc for batch mutation calls.

classmethod check_success(*response*, *msg*)

static dynamic_data(*mutations*)

Parameters **mutations** – list of mutation dictionaries

static_headers = {u'Content-Type': u'application/json'}

static_params = {u'alt': u'json'}

class gmusicapi.protocol.mobileclient.**McListCall**
Abc for calls that list a resource.

classmethod dynamic_data(*updated_after=None*, *start_token=None*, *max_results=None*)

Parameters

- **updated_after** – ignored
- **start_token** – nextPageToken from a previous response
- **max_results** – a positive int; if not provided, server defaults to 1000

classmethod dynamic_params(*updated_after=None*, *start_token=None*, *max_results=None*)

Parameters **updated_after** – datetime.datetime; defaults to epoch

classmethod parse_response(*response*)

static_headers = {u'Content-Type': u'application/json'}

static_params = {u'alt': u'json', u'include-tracks': u'true'}

class gmusicapi.protocol.mobileclient.**McStreamCall**

c1 = 71

c2 = 77

static dynamic_headers(*item_id*, *device_id*, *quality*)

classmethod dynamic_params(*item_id*, *device_id*, *quality*)

classmethod get_signature(*item_id*, *salt=None*)
Return a (sig, salt) pair for url signing.

static parse_response(*response*)

static_allow_redirects = False

class gmusicapi.protocol.mobileclient.**Search**
Search for All Access tracks.

static dynamic_params(*query*, *max_results*)

static_method = u'GET'

static_params = {u'ct': u'1,2,3,4,6,7,8,9'}

static_url = u'https://mclients.googleapis.com/sj/v2.5/query'

Music Manager

Calls made by the Music Manager (related to uploading).

class gmusicapi.protocol.musicmanager.**AuthenticateUploader**
Sent to auth, reauth, or register our upload client.

classmethod check_success(*response*, *msg*)

classmethod dynamic_data(*uploader_id*, *uploader_friendly_name*)

Parameters

- **uploader_id** – MM uses host MAC address

- **uploader_friendly_name** – MM uses hostname

static_url = u'https://android.clients.google.com/upsj/upauth'

class gmusicapi.protocol.musicmanager.**CancelUploadJobs**
This call will cancel any outstanding upload jobs (ie from GetJobs). The Music Manager only calls it when the user changes the location of their local collection.

It doesn't actually return anything useful.

static dynamic_data(*uploader_id*)

Parameters **uploader_id** – id

static_method = u'POST'

static_url = u'https://android.clients.google.com/upsj/deleteuploadrequested'

class gmusicapi.protocol.musicmanager.**DownloadTrack**
Given a url, retrieve a track. Unlike the Webclient, this requires authentication.

The entire Requests.Response is returned.

static dynamic_url(*url*)

Parameters **url** – result of a call to GetDownloadLink

classmethod parse_response(*response*)

static_method = u'GET'

class gmusicapi.protocol.musicmanager.**GetClientState**

classmethod dynamic_data(*uploader_id*)

Parameters **uploader_id** – MM uses host MAC address

static_url = u'https://android.clients.google.com/upsj/clientstate'

class gmusicapi.protocol.musicmanager.**GetDownloadLink**
Get a url where a track can be downloaded.

Auth is not needed to retrieve the resulting url.

static dynamic_headers(*sid*, *client_id*)

static dynamic_params(*sid*, *client_id*)

classmethod parse_response(*response*)

static_headers = {}

static_method = u'GET'

static_params = {u'version': 2}

static_url = u'https://music.google.com/music/export'

class gmusicapi.protocol.musicmanager.**GetUploadJobs**

classmethod check_success(*response*, *msg*)

classmethod dynamic_data(*uploader_id*)

Parameters **uploader_id** – MM uses host MAC address

static_params = {u'version': 1}

static_url = u'https://android.clients.google.com/upsj/getjobs'

class gmusicapi.protocol.musicmanager.**GetUploadSession**
Called when we want to upload; the server returns the url to use. This is a json call, and doesn't share much with the other calls.

static dynamic_data(*uploader_id*, *num_already_uploaded*, *track*, *filepath*, *server_id*, *do_not_rematch=False*)
track is a locker_pb2.Track, and the server_id is from a metadata upload.

classmethod parse_response(*response*)

static process_session(*res*)
Return (got_session, error_details). error_details is (should_retry, reason, error_code) or None if got_session.

static_method = u'POST'

static_url = u'https://uploadsj.clients.google.com/uploadsj/scottyagent'

class gmusicapi.protocol.musicmanager.**ListTracks**
List all tracks. Returns a subset of all available metadata. Can optionally filter for only free/purchased tracks.

classmethod check_success(*response*, *msg*)

static dynamic_data(*client_id*, *cont_token=None*, *export_type=1*, *updated_min=0*)
Works similarly to the webclient method. Chunks are up to 1000 tracks.

Parameters

- **client_id** – an authorized uploader_id
- **cont_token** – (optional) token to get the next library chunk.
- **export_type** – 1='ALL', 2='PURCHASED_AND_PROMOTIONAL'
- **updated_min** – likely a timestamp; never seen an example of this != 0

static dynamic_headers(*client_id*, **args*, ***kwargs*)

res_msg_type
alias of GetTracksToExportResponse

static_method = u'POST'

static_url = u'https://music.google.com/music/exportids'

class gmusicapi.protocol.musicmanager.**OAuthInfo**(*client_id*, *client_secret*, *scope*, *redirect*)

client_id
Alias for field number 0

client_secret
Alias for field number 1

redirect
Alias for field number 3

scope
Alias for field number 2

class gmusicapi.protocol.musicmanager.**ProvideSample**
Give the server a scan and match sample. The sample is a 128k mp3 slice of the file, usually 15 seconds long.

static dynamic_data(*filepath*, *server_challenge*, *track*, *uploader_id*, *mock_sample=None*)
Raise IOError on transcoding problems, or ValueError for invalid input.

Parameters **mock_sample** – if provided, will be sent in place of a proper sample

static_method = u'POST'

static_params = {u'version': 1}

static_url = u'https://android.clients.google.com/upsj/sample'

class gmusicapi.protocol.musicmanager.**UpdateUploadState**
Notify the server that we will be starting/stopping/pausing our upload.

I believe this is used for the webclient 'currently uploading' widget, but that might also be the current_uploading information.

static dynamic_data(*to_state*, *uploader_id*)
Raise ValueError on problems.

Parameters **to_state** – one of 'start', 'paused', or 'stopped'

static_method = u'POST'

static_params = {u'version': 1}

static_url = u'https://android.clients.google.com/upsj/uploadstate'

class gmusicapi.protocol.musicmanager.**UploadFile**
Called after getting a session to actually upload a file.

static dynamic_data(*session_url*, *content_type*, *audio*)

static dynamic_headers(*session_url*, *content_type*, *audio*)

static dynamic_url(*session_url*, *content_type*, *audio*)

classmethod parse_response(*response*)

static_method = u'PUT'

class gmusicapi.protocol.musicmanager.**UploadMetadata**

count_fields = {u'tracknumber': (u'track_number', u'total_track_count'), u'discnumber': (u'disc_number', u'total_

classmethod dynamic_data(*tracks*, *uploader_id*, *do_not_rematch=False*)

Parameters

- **tracks** – list of filled locker_pb2.Track
- **uploader_id** –
- **do_not_rematch** – seems to be ignored

field_map = {u'albumartist': u'album_artist', u'bpm': u'beats_per_minute'}

classmethod fill_track_info(*filepath*)
Given the path and contents of a track, return a filled locker_pb2.Track. On problems, raise ValueError.

static get_track_clientid(*filepath*)

shared_fields = (u'album', u'artist', u'composer', u'genre')

static_params = {u'version': 1}

static_url = u'https://android.clients.google.com/upsj/metadata'

gmusicapi.protocol.musicmanager.**credentials_from_refresh_token**(*token*)

gmusicapi.protocol.musicmanager.**pb**(*func*)
Decorator to serialize a protobuf message.

Web Client

Calls made by the web client.

class gmusicapi.protocol.webclient.**AddToPlaylist**
Adds songs to a playlist.

static dynamic_data(*playlist_id*, *song_ids*)

Parameters

- **playlist_id** – id of the playlist to add to.
- **song_ids** – a list of song ids

static_method = u'POST'

static_url = u'https://play.google.com/music/services/addtoplaylist'

class gmusicapi.protocol.webclient.**ChangePlaylistOrder**
Reorder existing tracks in a playlist.

static dynamic_data(*playlist_id*, *song_ids_moving*, *entry_ids_moving*, *after_entry_id=None*, *before_entry_id=None*)

Parameters

- **playlist_id** – id of the playlist getting reordered.
- **song_ids_moving** – a list of consecutive song ids. Matches entry_ids_moving.
- **entry_ids_moving** – a list of consecutive entry ids to move. Matches song_ids_moving.
- **after_entry_id** – the entry id to place these songs after. Default first position.
- **before_entry_id** – the entry id to place these songs before. Default last position.

static_method = u'POST'

static_url = u'https://play.google.com/music/services/changeplaylistorder'

class gmusicapi.protocol.webclient.**ChangeSongMetadata**
Edit the metadata of songs.

static dynamic_data(*songs*, *session_id=u''*)

Parameters **songs** – a list of dicts `{'id': '...', 'albumArtUrl': '...'}`

static_method = u'POST'

static_params = {u'format': u'jsarray'}

static_url = u'https://play.google.com/music/services/modifytracks'

class gmusicapi.protocol.webclient.**CreatePlaylist**
Adds songs to a playlist.

static dynamic_data(*name*, *description*, *public*, *session_id=u''*)

static_method = u'POST'

static_params = {u'format': u'jsarray'}

static_url = u'https://play.google.com/music/services/createplaylist'

class gmusicapi.protocol.webclient.**DeauthDevice**
Deauthorize a device from GetSettings.

static dynamic_data(*device_id*, *session_id*)

static_method = u'POST'

static_url = u'https://play.google.com/music/services/modifysettings'

class gmusicapi.protocol.webclient.**DeletePlaylist**
Delete a playlist.

static dynamic_data(*playlist_id*)

Parameters **playlist_id** – id of the playlist to delete.

static_method = u'POST'

static_url = u'https://play.google.com/music/services/deleteplaylist'

class gmusicapi.protocol.webclient.**DeleteSongs**
Delete a song from the entire library or a single playlist.

static dynamic_data(*song_ids*, *playlist_id=u'all'*, *entry_ids=None*)

Parameters

- **song_ids** – a list of song ids.
- **playlist_id** – playlist id to delete from, or 'all' for deleting from library.
- **entry_ids** – when deleting from playlists, corresponding list of entry ids.

static_method = u'POST'

static_url = u'https://play.google.com/music/services/deletesong'

class gmusicapi.protocol.webclient.**GetDownloadInfo**
Get download links and counts for songs.

static dynamic_data(*song_ids*)

Param (list) song_ids

static_method = u'POST'

static_url = u'https://play.google.com/music/services/multidownload'

class gmusicapi.protocol.webclient.**GetSettings**
Get data that populates the settings tab: labs and devices.

static dynamic_data(*session_id*)

Param session_id

static_method = u'POST'

static_url = u'https://play.google.com/music/services/fetchsettings'

class gmusicapi.protocol.webclient.**GetSharedPlaylist**
Get the contents and metadata for a shared playlist.

static dynamic_data(*session_id*, *share_token*)

classmethod parse_response(*response*)

static_method = u'POST'

static_params = {u'format': u'jsarray'}

static_url = u'https://play.google.com/music/services/loadsharedplaylist'

class gmusicapi.protocol.webclient.**GetStreamUrl**
Used to request a streaming link of a track.

static dynamic_params (*song_id*)

required_auth = AuthTypes(xt=False, sso=True, oauth=False)

static_method = u'GET'

static_url = u'https://play.google.com/music/play'

class gmusicapi.protocol.webclient.**Init**
Called one time per session, immediately after login.

This performs one-time setup: it gathers the cookies we need (specifically *xt*), and Google uses it to create the webclient DOM.

Note the use of the HEAD verb. Google uses GET, but we don't need the large response containing Google's webui.

classmethod check_success (*response*, *msg*)

static parse_response (*response*)

required_auth = AuthTypes(xt=False, sso=True, oauth=False)

static_method = u'HEAD'

static_url = u'https://play.google.com/music/listen'

class gmusicapi.protocol.webclient.**ReportBadSongMatch**
Request to signal the uploader to reupload a matched track.

static dynamic_data (*song_ids*)

static_method = u'POST'

static_params = {u'format': u'jsarray'}

static_url = u'https://play.google.com/music/services/fixsongmatch'

class gmusicapi.protocol.webclient.**UploadImage**
Upload an image for use as album art.

static dynamic_files (*image_filepath*)

Parameters **image_filepath** – path to an image

static_method = u'POST'

static_params = {u'u': 0, u'zx': u''}

static_url = u'https://play.google.com/music/services/imageupload'

2.6 Making gmusicapi better

Contributions are always welcome! The *contributing section* has more details.

The code might also be useful.

2.7 Ports and other languages

The *ports section* lists known ports and information for making ports.

2.8 Getting help

Running into bugs? Have questions? Drop by `#gmusicapi` on Freenode. If you've never used IRC before, it's easy: just fill in a nickname and captcha at this webchat link.

If IRC makes you uncomfortable, you can always email me directly: simon@simonmweber.com.

Python Module Index

g

Index

Symbols

A

B

C

D

I

L

M

O

P

U

W