Supysonic

Release 0.7.5

Alban Féron

Apr 02, 2023

CONTENTS

Supysonic is a Python implementation of the Subsonic server API.

Current supported features are:

- browsing (by folders or tags)
- streaming of various audio file formats
- transcoding
- user or random playlists
- cover art
- starred tracks/albums and ratings
- Last.FM scrobbling
- Jukebox mode

User's guide

CHAPTER

ONE

SUPYSONIC SETUP

This guide details the required steps to get a Supysonic instance ready to start serving your music.

TL;DR

For the impatient, here's a quick summary to get Supysonic installed and ready to start serving (but this doesn't create any user nor specifies where your music is located). This uses gunicorn, but there are *other options*.

```
pip install supysonic
pip install gunicorn
supysonic-server
```

Table of contents

1.1 Installing Supysonic

Supysonic is written in Python and supports Python 3.7 and later.

1.1.1 Linux

Currently, only Debian-based distributions might provide Supysonic in their package repositories. Install the package `supysonic` using **`apt`**:

```
$ apt install supysonic
```

This will install Supysonic along with the minimal dependencies it needs to run.

If you plan on using it with a MySQL or PostgreSQL database you also need the corresponding Python package, `python3-pymysql` for MySQL or `python3-psycopg2` for PostgreSQL.

```
$ apt install python3-pymysql
```

```
$ apt install python3-psycopg2
```

For other distributions, you might consider installing with *pip* or from *Docker* images.

1.1.2 Windows

Note: While Supysonic hasn't been thoroughly tested on Windows, it *should* work. If something is broken, we're really sorry. Don't hesitate to open an issue on GitHub.

Most Windows users do not have Python installed by default, so we begin with the installation of Python itself. To check if you already have Python installed, open the *Command Prompt* (`Win-R` and type **`cmd`**). Once the command prompt is open, type **`python --version`** and press Enter. If Python is installed, you will see the version of Python printed to the screen. If you do not have Python installed, head over to the Python website and install one of the compatible Python versions. You need at least Python 3.7.

Once Python is installed, you can install Supysonic using **`pip`**. Refer to the *installation instructions* below for more information.

1.1.3 pip

Simply install the package `supysonic` with **`pip`**:

```
$ pip install supysonic
```

This will install Supysonic along with the minimal dependencies it needs, but those don't include the requirements for the web server. For this you'll need to install either `gevent`, `gunicorn` or `waitress`.

```
$ pip install gevent
```

```
$ pip install gunicorn
```

```
$ pip install waitress
```

If you plan on using it with a MySQL or PostgreSQL database you also need the corresponding package, `pymysql` for MySQL or `psycopg2-binary` for PostgreSQL.

```
$ pip install pymysql
```

```
$ pip install psycopg2-binary
```

1.2 Database setup

Supysonic needs a database to run. It can either be a SQLite, MySQL-compatible or PostgreSQL database.

If you absolutely have no clue about databases, you can go with SQLite as it doesn't need any setup other than specifying a path for the database in the *configuration*.

Note: SQLite, while being a viable option, isn't recommended for large installations. First of all its performance *might* start to decrease as the size of your library grows. But most importantly if you have a lot of users reaching the instance at the same time you will start to see the performance drop, or even errors.

Please refer to the documentation of the DBMS you've chosen on how to create a database. Once it has a database, Supysonic will automatically create the tables it needs and keep the schema up-to-date.

1.2.1 The PostgreSQL case

If you want to use PostgreSQL you'll have to add the `citext` extension to the database once created. This can be done when connected to the database as the superuser. How to connect as a superuser might change depending on your PostgreSQL installation (this is **not** the same thing as the OS superuser known as *root* on Linux systems).

On a Debian-based system you can connect as a superuser by invoking **`psql`** while being logged in as the *postgres* user. The following commands will install the `citext` extension on the database named *supysonic* assuming you are currently logged as *root*.

```
# su - postgres
$ psql supysonic
supysonic=# CREATE EXTENSION citext;
```

1.3 Configuration

Supysonic looks for four files for its configuration: `/etc/supysonic`, `~/.supysonic`, `~/.config/supysonic/supysonic.conf` and `supysonic.conf` in the current working directory, in this order, merging values from all files.

Configuration files must respect a structure similar to Windows INI file, with `[section]` headers and using a `KEY = VALUE` or `KEY: VALUE` syntax.

If you cloned Supysonic from its GitHub repository you'll find a roughly documented configuration sample file at the root of the project, file conveniently named `config.sample`. More details below.

1.3.1 `[base]` section

This sections defines the database and additional scanning config.

`database_uri`
: The most important configuration, defines the type and parameters of the database Supysonic should connect to. It usually includes username, password, hostname and database name. The typical form of a database URI is:

  ```
  driver://username:password@host:port/database
  ```

  If the connection needs some additional parameters, they can be provided as a query string, such as:

  ```
  driver://username:password@host:port/database?param1=value1&param2=value2
  ```

  Supported drivers are `sqlite`, `mysql` and `postgres` (or `postgresql`).

  As SQLite connects to local files, the format is slightly different. The "file" portion of the URI is the filename of the database. For a relative path, it requires three slashes, for absolute paths it's also three slashes followed by the absolute path, meaning actually four slashes on Unix systems.

  ```
  ; Relative path
  database_uri = sqlite:///relative-file.db
  ; Absolute path on Unix-based systems
  database_uri = sqlite:////home/user/supysonic.db
  ; Absolute path on Windows
  database_uri = sqlite:///C:\Users\user\supysonic.db
  ```

  A MySQL-compatible database requires either `MySQLdb` or `pymysql` to be installed. PostgreSQL needs `psycopg2`.

Note: For MySQL if no character set is defined on the URI it defaults to `utf8mb4` regardless of what's set on your MySQL installation.

If `database_uri` isn't provided, it defaults to a SQLite database stored in `/tmp/supysonic/supysonic.db`.

scanner_extensions
A space separated list of file extensions the scanner is restricted to. Useful if you have multiple audio formats in your library but only want to serve some. If left empty, the scanner will try to read every file it finds.

follow_symlinks
If set to `yes`, allows the scanner to follow symbolic links.

Disabled by default, enable it only if you trust your file system as nothing is done to handle broken links or loops.

Sample configuration:

```
[base]
; A database URI. Default: sqlite:////tmp/supysonic/supysonic.db
database_uri = sqlite:////var/supysonic/supysonic.db
;database_uri = mysql://supysonic:supysonic@localhost/supysonic
;database_uri = postgres://supysonic:supysonic@localhost/supysonic

; Optional, restrict scanner to these extensions. Default: none
scanner_extensions = mp3 ogg

; Should the scanner follow symbolic links? Default: no
follow_symlinks = no
```

1.3.2 [webapp] section

Configuration relative to the HTTP server.

cache_dir
Directory used to store generated files, such as resized cover art or transcoded files. Defaults to `/tmp/supysonic`.

cache_size
Maximum size (in megabytes) of the cache (except for trancodes). Defaults to 512 MB.

transcode_cache_size
Maximum size (in megabytes) of the transcode cache. Defaults to 1024 MB (1 GB).

log_file
Rotating file where some events generated by the web server are logged. Leave empty to disable logging.

log_level
Defines the minimum severity threshold of messages to be added to `log_file`. Possible values are:

- `DEBUG`
- `INFO`
- `WARNING`
- `ERROR`
- `CRITICAL`

Defaults to `WARNING`.

log_rotate
Enable automatic log rotation (when logs are enabled) every day at midnight. Set it to no if you don't want to rotate the logs or if you use external utilities such as **logrotate**. Defaults to yes.

mount_api (on or off)
Enable or disable the Subsonic REST API. Should be kept on or Supysonic would be quite useless. Exists mostly for testing purposes. Defaults to on.

mount_webui (on or off)
Enable or disable the administrative web interface.

> **Note:** Setting this off will prevent users from defining a preferred transcoding format.

Defaults to on.

index_ignored_prefixes
Space-separated list of prefixes that should be ignored from artist names when returning their index. Example: if the word *The* is in this list, artist *The Rolling Stones* will be listed under the letter *R*. The match is case insensitive. Defaults to `El La Le Las Les Los The`.

online_lyrics
If enabled, will fetch the lyrics (when requested) from ChartLyrics if they aren't available locally (either from metadata or from text files). Defaults to no.

Sample configuration:

```
[webapp]
; Optional cache directory. Default: /tmp/supysonic
cache_dir = /var/supysonic/cache

; Main cache max size in MB. Default: 512
cache_size = 512

; Transcode cache max size in MB. Default: 1024 (1GB)
transcode_cache_size = 1024

; Optional rotating log file. Default: none
log_file = /var/supysonic/supysonic.log

; Log level. Possible values: DEBUG, INFO, WARNING, ERROR, CRITICAL.
; Default: WARNING
log_level = WARNING

; Enable log rotation. Default: yes
log_rotate = yes

; Enable the Subsonic REST API. You'll most likely want to keep this on.
; Here for testing purposes. Default: on
;mount_api = on

; Enable the administrative web interface. Default: on
;mount_webui = on

; Space separated list of prefixes that should be ignored on index endpoints
; Default: El La Le Las Les Los The
```

(continues on next page)

(continued from previous page)

```
index_ignored_prefixes = El La Le Las Les Los The

; Enable the ChartLyrics API. Default: off
online_lyrics = off
```

1.3.3 [daemon] section

Configuration for the daemon process that is used to watch for changes in the library folders and providing the jukebox feature.

socket
: Unix domain socket file (or named pipe on Windows) used to communicate between the daemon and clients that rely on it (eg. CLI, folder admin web page, etc.). Note that using an IP address here isn't supported. Default: `/tmp/supysonic/supysonic.sock`

run_watcher
: Whether or not to start the watcher that will listen for library changes. Default: yes

wait_delay
: Delay (in seconds) before triggering the scanning operation after a change have been detected. This prevents running too many scans when multiple changes are detected for a single file over a short time span. Default: 5 seconds.

jukebox_command
: Command used by the jukebox mode to play a single file. See the *jukebox documentation* for more details.

log_file
: Rotating file where events generated by the file watcher are logged. If left empty, any logging will be sent to stderr.

log_level
: Defines the minimum severity threshold of messages to be added to `log_file`. Possible values are:

  - `DEBUG`
  - `INFO`
  - `WARNING`
  - `ERROR`
  - `CRITICAL`

  Defaults to `WARNING`.

log_rotate
: Enable automatic log rotation (when logs are enabled) every day at midnight. Set it to `no` if you don't want to rotate the logs or if you use external utilities such as **`logrotate`**. Defaults to `yes`.

Sample configuration:

```
[daemon]
; Socket file the daemon will listen on for incoming management commands
; Default: /tmp/supysonic/supysonic.sock
socket = /var/run/supysonic.sock
; Syntax for windows named pipe:
;socket = \\.\pipe\supysonic.sock
```

(continues on next page)

(continued from previous page)

```
; Defines if the file watcher should be started. Default: yes
run_watcher = yes

; Delay in seconds before triggering scanning operation after a change have been
; detected.
; This prevents running too many scans when multiple changes are detected for a
; single file over a short time span. Default: 5
wait_delay = 5

; Command used by the jukebox
jukebox_command = mplayer -ss %offset %path

; Optional rotating log file for the scanner daemon. Logs to stderr if empty
log_file = /var/supysonic/supysonic-daemon.log
log_level = INFO

; Enable log rotation. Default: yes
log_rotate = yes
```

1.3.4 `[lastfm]` section

This section allow defining API keys to enable Last.FM integration in Supysonic. Currently it is only used to *scrobble* played tracks and update the *now playing* information.

See https://www.last.fm/api to obtain such keys.

Once keys are set, users have to link their account by visiting their profile page on Supysonic's administrative UI.

api_key
: Last.FM API key

secret
: secret key associated to the API key

Sample configuration:

```
[lastfm]
; API and secret key to enable scrobbling. http://www.last.fm/api/accounts
; Defaults: none
;api_key =
;secret =
```

1.3.5 `[transcoding]` section

This section defines command-line programs to be used to convert an audio file to another format or change its bitrate. All configurations in the sample below have **not** been thoroughly tested. For more details, please refer to the *transcoding configuration*.

```
[transcoding]
; Programs used to convert from one format/bitrate to another. Defaults: none
transcoder_mp3_mp3 = lame --quiet --mp3input -b %outrate %srcpath -
transcoder = ffmpeg -i %srcpath -ab %outratek -v 0 -f %outfmt -
```

(continues on next page)

(continued from previous page)

```
decoder_mp3 = mpg123 --quiet -w - %srcpath
decoder_ogg = oggdec -o %srcpath
decoder_flac = flac -d -c -s %srcpath
encoder_mp3 = lame --quiet -b %outrate - -
encoder_ogg = oggenc2 -q -M %outrate -
```

1.3.6 [mimetypes] section

Use this section if the system Supysonic is installed on has trouble guessing the mimetype of some files. This might only be useful in some rare cases.

See the following links for a list of examples:

- https://en.wikipedia.org/wiki/Media_type#Common_examples
- https://www.iana.org/assignments/media-types/media-types.xhtml

```
[mimetypes]
; Extension to mimetype mappings in case your system has some trouble guessing
; Default: none
;mp3 = audio/mpeg
;ogg = audio/vorbis
```

1.4 Running the web server

Once Supysonic is installed and configured, you'll have to start its web server for the clients to be able to access the music. Here you have several options, whether you want to run it as independant process(es), then possibly putting it behind a reverse proxy, or running it as a WSGI application within Apache.

1.4.1 supysonic-server

But the easiest might be to use Supysonic's own server. It actually requires a WSGI server library to run, so you'll first need to have either Gevent, Gunicorn or Waitress to be installed. Then you can start the server with the following command:

```
supysonic-server
```

And it will start to listen on all IPv4 interfaces on port 5722.

This command allows some options, more details are given on its manpage: *supysonic-server*. It is intentionally kept simple, as such it doesn't provide much in terms of tuning. If you want more control over the server's behavior you might as well try one of the options presentend below.

1.4.2 Other options

You'll find some other common (and less common) deployment options below:

Standalone WSGI Containers

There are popular servers written in Python that contain WSGI applications and serve HTTP. These servers stand alone when they run; you can let your clients access them directly or proxy to them from your web server such as Apache or nginx.

Gunicorn

Gunicorn "Green Unicorn" is a WSGI HTTP Server for UNIX. It's a pre-fork worker model. Running Supysonic on this server is quite simple. First install Gunicorn with either **`pip install gunicorn`** or **`apt install gunicorn3`** (the `gunicorn` package in this case is for Python 2 which isn't supported anymore). Then:

```
$ gunicorn "supysonic.web:create_application()"
```

But this will only listen on the loopback interface, which isn't really useful.

Gunicorn provides many command-line options -- see **`gunicorn -h`**. For example, to run Supysonic with 4 worker processes (`-w 4`) binding to all IPv4 interfaces on port 5722 (`-b 0.0.0.0:5722`):

```
$ gunicorn -w 4 -b 0.0.0.0:5722 "supysonic.web:create_application()"
```

Note: While **`gunicorn`** provides way more options to configure its behaviour than **`supysonic-server`** will ever do, the above example is actually equivalent to:

```
$ supysonic-server -S gunicorn --processes 4
```

---

uWSGI

uWSGI is a fast application server written in C. It is very configurable which makes it more complicated to setup than Gunicorn.

To use it, install the package `uwsgi` with either **`pip`** or **`apt`**. Using the later, wou might also need the additional package `uwsgi-plugin-python3`.

Then to run Supysonic in uWSGI:

```
$ uwsgi --http-socket 0.0.0.0:5722 --module "supysonic.web:create_application()"
```

If it complains about an unknown `--module` option, try adding `--plugin python3`:

```
$ uwsgi --http-socket 0.0.0.0:5722 --plugin python3 --module "supysonic.web:create_
↪application()"
```

As uWSGI is highly configurable there are several options you could use to tweak it to your liking. Detailing all it can do is way beyond the scope of this documentation, if you're interested please refer to its documentation.

If you plan on using uWSGI behind a nginx reverse proxy, note that nginx provides options to integrate directly with uWSGI. You'll find an example configuration in Flask's documentation (the framework Supysonic is built upon). Replace the `myapp:app` in their example by `supysonic.web:create_application()` (you might need to enclose it in double-quotes).

Waitress

Waitress is meant to be a production-quality pure-Python WSGI server with very acceptable performance. It has no dependencies except ones which live in the Python standard library.

As for Gunicorn, using it to run Supysonic is rather simple. Install it using either **`pip install waitress`** or **`apt install python3-waitress`**. Then start the server this way:

```
$ waitress-serve --call supysonic.web:create_application
```

Waitress behaviour can be tuned through various command-line options -- see **`waitress-serve --help`**. If none of them are relevant to you, **`supysonic-server`** can actually be used instead:

```
$ supysonic-server -S waitress
```

Both commands are equivalent, with the only difference being the port they listen on.

Apache and mod_wsgi

If you are using the Apache webserver, you can use it to run Supysonic with the help of mod_wsgi.

Installing *mod_wsgi*

If you don't have *mod_wsgi* installed yet you have to install it and enable it first as follows:

```
# apt install libapache2-mod-wsgi-py3
# a2enmod wsgi
```

Creating a *.wsgi* file

To run Supysonic within Apache you need a `supysonic.wsgi` file. Create one somewhere and fill it with the following content:

```
from supysonic.web import create_application
application = create_application()
```

Store that file somewhere that you will find it again (e.g.: `/var/www/supysonic/supysonic.wsgi`).

Configuring Apache

The last thing you have to do is to edit the Apache configuration to tell it to load the application. Here's a basic example of what it looks like:

```
WSGIScriptAlias /supysonic /var/www/supysonic/supysonic.wsgi
<Directory /var/www/supysonic>
   WSGIApplicationGroup %{GLOBAL}
   WSGIPassAuthorization On
   Require all granted
</Directory>
```

With that kind of configuration, the server address will look like *http://server/supysonic/*.

For more information consult the mod_wsgi documentation. Note that the `WSGIPassAuthorization` directive is required for some clients as they provide their credentials using the *basic access authentification* mechanism rather than as URL query parameters.

Other options

FastCGI

FastCGI is a deployment option on servers like nginx or lighttpd; see *Standalone WSGI Containers* for other options. To use Supysonic with any of them you will need a FastCGI server first. The most popular one is flup which we will use for this guide. Make sure to have it installed (with eith **`pip`** or **`apt`**) to follow along.

Creating a *.fcgi* file

First you need to create the FastCGI server file. Let's call it `supysonic.fcgi`:

```
#!/usr/bin/python3

from flup.server.fcgi import WSGIServer
from supysonic.web import create_application

if __name__ == "__main__":
   app = create_application()
   WSGIServer(app).run()
```

This should be enough for Apache to work, however nginx and older versions of lighttpd need a socket to be explicitly passed to communicate with the FastCGI server. For that to work you need to pass the path to the socket to the `WSGIServer`:

```
WSGIServer(app, bindAddress="/path/to/fcgi.sock").run()
```

The path has to be the exact same path you define in the server config.

Save the `supysonic.fcgi` file somewhere you will find it again. It makes sense to have that in `/var/www/supysonic` or something similar.

Make sure to set the executable bit on that file so that the servers can execute it:

```
$ chmod +x /var/www/supysonic/supysonic.fcgi
```

Configuring the web server

The example above is good enough for a basic Apache deployment but your *.fcgi* file will appear in your application URL e.g. `example.com/supysonic.fcgi/`. If that bothers you or you wish to load it in another web server, Flask's documentation details how to do it for Apache, lighttpd or nginx.

CGI

If all other deployment methods do not work, CGI will work for sure. CGI is supported by all major servers but usually has a sub-optimal performance.

Creating a *.cgi* file

First you need to create the CGI application file. Let's call it `supysonic.cgi`:

```
#!/usr/bin/python3

from wsgiref.handlers import CGIHandler
from supysonic.web import create_application

app = create_application()
CGIHandler().run(app)
```

Server Setup

Usually there are two ways to configure the server. Either just copy the `.cgi` into a `cgi-bin` (and use *mod_rewrite* or something similar to rewrite the URL) or let the server point to the file directly.

In Apache for example you can put something like this into the config:

```
ScriptAlias /supysonic /path/to/the/supysonic.cgi
```

As Supysonic is a WSGI application, you have numerous deployment options available to you. If you want to deploy it to a WSGI server not listed here, look up the server documentation about how to use a WSGI app with it. When setting one of those, you'll want to call the `create_application()` factory function from module `supysonic.web`.

1.5 The daemon

Supysonic comes with an optional daemon service that currently provides the following features:

- background scans
- library changes detection
- jukebox mode

1.5.1 Background scans

First of all, the daemon allows running backgrounds scans, meaning you can start scans from the *command-line interface* and do something else while it's scanning (otherwise the scan will block the CLI until it's done). Background scans also enable the web UI to run scans, while you have to use the CLI to do so if you don't run the daemon.

1.5.2 Library watching

Instead of manually running a scan every time your library changes, the daemon can listen to any library change and update the database accordingly. This watcher is started along with the daemon but can be disabled to only keep background scans. Please refer to *[daemon] section* of the configuration to enable or disable it.

1.5.3 Jukebox

Finally, the daemon acts as a backend for the jukebox mode, allowing to play audio on the machine running Supysonic. More details on the *Jukebox mode* page.

1.5.4 Running it

The daemon is *supysonic-daemon*, it is a non-exiting process. If you want to keep it running in background, either use the old **nohup** or **screen** methods, or start it as a systemd unit.

Below is a basic service file to load it through systemd. Modify it to match your installation and save it as `/etc/systemd/system/supysonic-daemon.service`.

```
[Unit]
Description=Supysonic Daemon

[Service]
User=someuser
Group=somegroup
WorkingDirectory=/home/supysonic
ExecStart=/usr/bin/python3 -m supysonic.daemon

[Install]
WantedBy=multi-user.target
```

Docker

Another solution rather than going through the whole setup process yourself is to use a ready-to-use Docker image. While we don't provide images for Supysonic, that didn't keep the community from creating some. Take a look on the Docker Hub and pick one you like. For more details on their usage, please refer to the readme of said images.

CHAPTER

TWO

USING SUPYSONIC

Now that everything is *set up*, there's actually some more administrative tasks to perform before really being able to access your music. The first one being *Adding users* and next *Defining and scanning folders*. Then you can choose one of the many *Clients* and you're good to go.

2.1 Adding users

One of the first thing you want to do is to create the user(s) that will be allowed to access Supysonic. The first user has to be created with the CLI (command-line interface) (*manpage here*), if you set them as an admin this new user will then be able to add more users through *The web interface*.

Creating a new user and giving them administrative rights is done with the following two commands:

```
$ supysonic-cli user add TheUserName
$ supysonic-cli user setroles --admin TheUserName
```

The first command will ask for a password but you can also provide it on the command-line:

```
$ supysonic-cli user add TheUserName --password ThePassword
```

If you don't want to set the user as an admin but still want them to be able to use the *Jukebox mode*, you can give them the right like so:

```
$ supysonic-cli user setroles --jukebox TheUserName
```

This last one isn't needed for admins as they have full control over the installation.

2.2 Defining and scanning folders

Supysonic will be pretty useless if you don't tell it where your music is located. This can once again be done with the CLI or the web interface.

Using the CLI:

```
$ supysonic-cli folder add SomeFolderName /path/where/the/music/is
```

The next step is now to scan the folder to find all the media files it holds:

```
$ supysonic-cli folder scan SomeFolderName
```

If *The daemon* is running, this will start scanning in the background, otherwise you'll have to wait for the scan to end. This can take some time if you have a huge library.

2.3 The web interface

Once you have created a user, you can access the web interface at the root URL where the application is deployed. As Supysonic is mostly a server and not a media player this interface won't provide much. It is mainly used for administrative purposes but also provides some features for regular users that are only available through this interface.

Once logged, users can click on their username in the top bar to access some settings. These include the ability to link their Last.fm account provided Supysonic was *configured* with Last.fm API keys. Once linked clients will then be able to send *scrobbles*.

Note: In the case of Android clients (this haven't been tested with iOS) this could lead to *scrobbles* being sent twice if the official Last.fm application is also installed on the device.

Another setting also available only through the web interface is the ability to define the *preferred transcoding format*.

Admins got two more options accessible from the top bar: the ability to manage users and folders. But these have limitations compared to the CLI: you can't grant or revoke the users' jukebox privilege and you can scan folders you added only if *The daemon* is running.

2.4 Clients

You'll need a client to access your music. Whether you want an app for your smartphone, something running on your desktop or in a web page you got several options here.

One good start would be looking at the list on Subsonic website but that list *could* be a bit out of date and there's also some players that don't appear here. Also disregard the trial notice there, Supysonic doesn't include such nonsense.

Here are some hand-picked clients:

- in your browser:
  - SubPlayer (source, especially designed to work with Supysonic)
  - Jamstash (source, whose maintainer contributed to Supysonic)
- on Android:
  - Ultrasonic (source, whose maintainer contributed to Supysonic)
  - DSub (source)
- on iOS device:
  - you'll have to find one yourself
- for the desktop (none of them were tested)
  - Clementine
  - MusicBee with a plugin

Note: The Subsonic API provides several authentication methods. One of them, known as *token authentication* was added with API version 1.13.0. As Supysonic currently targets API version 1.9.0, the token based method isn't supported. So if your client offers you the option, you'll have to disable the token based authentication for it to work.

CHAPTER

THREE

TRANSCODING

Transcoding is the process of converting from one audio format to another. This allows for streaming of formats that wouldn't be streamable otherwise, or reducing the quality of an audio file to allow a decent streaming for clients with limited bandwidth, such as the ones running on a mobile connection.

Transcoding in Supysonic is achieved through the use of third-party command-line programs. Supysonic isn't bundled with such programs, and you are left to choose which one you want to use.

If you want to use transcoding but your client doesn't allow you to do so, you can force Supysonic to transcode for that client by going to your profile page on the web interface.

3.1 Configuration

Configuration of transcoders is done on the *[transcoding] section* of the configuration file.

Transcoding can be done by one single program which is able to convert from one format directly to another one, or by two programs: a decoder and an encoder. All these are defined by the following variables:

- `transcoder_EXT_EXT`
- `decoder_EXT`
- `encoder_EXT`
- `trancoder`
- `decoder`
- `encoder`
- `default_transcode_target`

where `EXT` is the lowercase file extension of the matching audio format. `transcoder`s variables have two extensions: the first one is the source extension, and the second one is the extension to convert to. The same way, `decoder`s extension is the source extension, and `encoder`s extension is the extension to convert to. The value of `default_transcode_target` will be used as output format when a client requests a bitrate lower than the original file and no specific format.

Notice that all of them have a version without extension. Those are generic versions. The programs defined with these variables should be able to transcode/decode/encode any format. For that reason, we suggest you don't use these if you want to keep control over the available transcoders.

Supysonic will take the first available transcoding configuration in the following order:

1. specific transcoder
2. specific decoder / specific encoder

3. generic decoder / generic encoder (with the possibility to use a generic decoder with a specific encoder, and vice-versa)
4. generic transcoder

All the variables should be set to the command-line used to run the converter program. The command-lines can include the following fields:

%srcpath
path to the original file to transcode

%srcfmt
extension of the original file

%outfmt
extension of the resulting file

%outrate
bitrate of the resulting file

%title
title of the file to transcode

%album
album name of the file to transcode

%artist
artist name of the file to transcode

%tracknumber
track number of the file to transcode

%totaltracks
number of tracks in the album of the file to transcode

%discnumber
disc number of the file to transcode

%genre
genre of the file to transcode (not always available, defaults to "")

%year
year of the file to transcode (not always available, defaults to "")

One final note: the original file should be provided as an argument of transcoders and decoders. All transcoders, decoders and encoders should write to standard output, and encoders should read from standard input (decoders output being piped into encoders)

3.1.1 Suggested configuration

Here is an example configuration that you could use. This is provided as-is, and some configurations haven't been tested.

Basic configuration:

```
[transcoding]
transcoder_mp3_mp3 = lame --quiet --mp3input -b %outrate %srcpath -
transcoder = ffmpeg -i %srcpath -ab %outratek -v 0 -f %outfmt -
decoder_mp3 = mpg123 --quiet -w - %srcpath
decoder_ogg = oggdec -o %srcpath
```

(continues on next page)

(continued from previous page)

```
decoder_flac = flac -d -c -s %srcpath
encoder_mp3 = lame --quiet -b %outrate - -
encoder_ogg = oggenc2 -Q -M %outrate -
default_transcode_target = mp3
```

To include track metadata in the transcoded stream:

```
[transcoding]
transcoder_mp3_mp3 = lame --quiet --mp3input -b %outrate --tt %title --tl %album --ta
↪%artist --tn %tracknumber/%totaltracks --tv TPOS=%discnumber --tg %genre --ty %year --
↪add-id3v2 %srcpath -
transcoder = ffmpeg -i %srcpath -ab %outratek -v 0 -metadata title=%title -metadata␣
↪album=%album -metadata author=%artist -metadata track=%tracknumber/%totaltracks -
↪metadata disc=%discnumber -metadata genre=%genre -metadata date=%year -f %outfmt -
decoder_mp3 = mpg123 --quiet -w - %srcpath
decoder_ogg = oggdec -o %srcpath
decoder_flac = flac -d -c -s %srcpath
encoder_mp3 = lame --quiet -b %outrate --tt %title --tl %album --ta %artist --tn
↪%tracknumber/%totaltracks --tv TPOS=%discnumber --tg %genre --ty %year --add-id3v2 - -
encoder_ogg = oggenc2 -Q -M %outrate -t %title -l %album -a %artist -N %tracknumber -c␣
↪TOTALTRACKS=%totaltracks -c DISCNUMBER=%discnumber -G %genre -d %year -
default_transcode_target = mp3
```

3.2 Enabling transcoding

Once the transcoding configuration has been set, most clients will require the user to specify that they want to transcode files. This might be done on the client itself, but most importantly it should be done on Supysonic web interface. Not doing so might prevent some clients to properly request transcoding.

To enable transcoding with the web interface, you should first start using the client you want to set transcoding for. Only browsing the library should suffice. Then open your browser of choice and navigate to the URL of your Supysonic instance. Log in with your credentials and the click on your username in the top bar. There you should be presented with a list of clients you used to connect to Supysonic and be able to set your preferred streaming format and bitrate.

CHAPTER

FOUR

JUKEBOX MODE

The jukebox mode allow playing audio files on the hardware of the machine running Supysonic, using regular clients that support it as a remote control.

The daemon must be running in order to be able to use the jukebox mode. So be sure to start the *supysonic-daemon* command and keep it running.

4.1 Setting the player program

Jukebox mode in Supysonic works through the use of third-party command-line programs. Supysonic isn't bundled with such programs, and you are left to choose which one you want to use. The chosen program should be able to play a single audio file from a path specified on its command-line.

The configuration is done in the *[daemon] section* of the configuration file, with the `jukebox_command` variable. This variable should include the following fields:

`%path`
: absolute path of the file to be played

`%offset`
: time in seconds where to start playing (used for seeking)

Here's an example using `mplayer`:

```
jukebox_command = mplayer -ss %offset %path
```

Or using `mpv`:

```
jukebox_command = mpv --start=%offset %path
```

Setting the output volume isn't currently supported.

4.2 Allowing users to act on the jukebox

The jukebox mode is only accessible to chosen users. Granting (or revoking) jukebox usage rights to a specific user is done with the *command line interface*:

```
$ supysonic-cli user setroles --jukebox <username>
```

CHAPTER

FIVE

MAN PAGES

Command-line interface

5.1 supysonic-cli

5.1.1 SYNOPSIS

supysonic-cli *--help*

supysonic-cli **user** [*options*]

supysonic-cli **folder** [*options*]

5.1.2 DESCRIPTION

Supysonic is a Python implementation of the Subsonic server API. Current supported features are:

- browsing (by folders or tags)
- streaming of various audio file formats
- transcoding
- user or random playlists
- cover arts (as image files in the same folder as music files)
- starred tracks/albums and ratings
- Last.FM scrobbling
- Jukebox mode

The "Subsonic API" is a set of adhoc standards to browse, stream or download a music collection over HTTP.

The command-line interface is an interface allowing administration operations without the use of the web interface.

5.1.3 SUBCOMMANDS

supysonic-cli has two different subcommands:

user [*options*]
User management commands

folder [*options*]
Folder management commands

For more details on the **user** and **folder** subcommands, see the `subsonic-cli-user (1)`, `subsonic-cli-folder (1)` manual pages.

5.1.4 OPTIONS

-h, --help
Shows the help and exits. At top level it only lists the subcommands. To display the help of a specific subcommand, add the **--help** flag *after* the said subcommand name.

5.1.5 BUGS

Bugs can be reported to your distribution's bug tracker or upstream at https://github.com/spl0k/supysonic/issues.

5.1.6 SEE ALSO

`supysonic-cli-user (1)`, `supysonic-cli-folder (1)`, `supysonic-server (1)`, `supysonic-daemon (1)`

5.2 supysonic-cli-user

5.2.1 SYNOPSIS

supysonic-cli user *--help*

supysonic-cli user **list**

supysonic-cli user **add** <*user*> [*--password* <*password*>] [*--email* <*email*>]

supysonic-cli user **delete** <*user*>

supysonic-cli user **changepass** <*user*> [*--password* <*password*>]

supysonic-cli user **setroles** [*--admin* | *--noadmin*] [*--jukebox* | *--nojukebox*] <*user*>

supysonic-cli user **rename** <*user*> <*newname*>

5.2.2 DESCRIPTION

The **supysonic-cli user** subcommand manages users, allowing to list them, add a new user, delete an existing user, and change their password or roles.

5.2.3 ARGUMENTS

list
List all the users.

add <*user*> [*--password* <*password*>] [*--email* <*email*>]
Add a new user named <*user*>. Will prompt for a password if it isn't given with the *--password* option.

delete <*user*>
Delete the user <*user*>.

changepass <*user*> [*--password* <*password*>]
Change the password of user <*user*>. Will prompt for the new password if not provided.

setroles [*--admin* | *--noadmin*] [*--jukebox* | *--nojukebox*] <*user*>
Give or remove rights to user <*user*>.

rename <*user*> <*newname*>
Rename the user <*user*> to <*newname*>.

5.2.4 OPTIONS

-h, --help
Shows help and exits. Depending on where this option appears it will either list the available commands or display help for a specific command.

-p <*password*>, --password <*password*>
Specify the user's password upon creation.

-e <*email*>, --email <*email*>
Specify the user's email.

The next options relate to user roles. They work in pairs, one option granting a right while the other revokes it; obviously options of the same pair are mutually exclusive.

The long options are named with the matching right, prefix it with a **no** to revoke the right. For short options, the upper case letter grants the right while the lower case letter revokes it. Short options might be combined into a single one such as **-aJ** to both revoke the admin right and grant the jukebox one.

-A, --admin
Grant admin rights.

-a, --noadmin
Revoke admin rights.

-J, --jukebox
Grant jukebox rights.

-j, --nojukebox
Revoke jukebox rights.

5.2.5 EXAMPLES

To add a new admin user named `MyUserName` having password `MyAwesomePassword`:

```
$ supysonic-cli user add MyUserName -p MyAwesomePassword
$ supysonic-cli user setroles -A MyUserName
```

5.2.6 SEE ALSO

`supysonic-cli (1)`, `supysonic-cli-folder (1)`, `supysonic-server (1)`, `supysonic-daemon (1)`

5.3 supysonic-cli-folder

5.3.1 SYNOPSIS

supysonic-cli folder *--help*

supysonic-cli folder **list**

supysonic-cli folder **add** <*name*> <*path*>

supysonic-cli folder **delete** <*name*>

supysonic-cli folder **scan** [*--force*] [*--background* | *--foreground*] <*name*>

5.3.2 DESCRIPTION

The **supysonic-cli folder** subcommand manages your library folders, where the audio files are located. This allows one to list, add, delete and scan the folders.

5.3.3 ARGUMENTS

list
: List all the folders.

add <*name*> <*path*>
: Add a new library folder called <*name*> and located at <*path*>. <*name*> must be unique and <*path*> pointing to an existing directory. If `supysonic-daemon` is running it will start to listen for changes in this folder but will not scan files already present in the folder.

delete <*name*>
: Delete the folder called <*name*>.

scan [*--force*] [*--background* | *--foreground*] <*name*>
: Scan the specified folders. If none is given, all the registered folders are scanned.

5.3.4 OPTIONS

-h, --help
Shows help and exits. Depending on where this option appears it will either list the available commands or display help for a specific command.

-f, --force
Force scan of already known files even if they haven't changed. Might be useful if an update to supysonic adds new metadata to audio files.

--background
Scan in the background. Requires the `supysonic-daemon` to be running.

--foreground
Scan in the foreground, blocking the process while the scan is running.

If neither **--background** nor **--foreground** is provided, supysonic-cli will try to connect to the daemon to initiate a background scan, falling back to a foreground scan if it isn't available.

5.3.5 EXAMPLES

To add a new folder to your music library, you can do something like this:

```
$ supysonic-cli folder add MyLibrary /home/username/Music
```

Once you've added a folder, you will need to scan it:

```
$ supysonic-cli folder scan MyLibrary
```

The audio files residing in `/home/username/Music` will now appear under the `MyLibrary` folder on the clients.

5.3.6 SEE ALSO

`supysonic-cli (1)`, `supysonic-cli-user (1)`, `supysonic-server (1)`, `supysonic-daemon (1)`

Web server

5.4 supysonic-server

5.4.1 SYNOPSIS

supysonic-server [**--server** *gevent* | *gunicorn* | *waitress*] [**--host** <*hostname*>] [**--port** <*port*>] [**--socket** <*path*>] [**--processes** <*n*>] [**--threads** <*n*>]

5.4.2 DESCRIPTION

supysonic-server is the main supysonic's component, allowing to serve content to clients. It is actually a basic wrapper over **Gevent**, **Gunicorn** or **Waitress**, requiring at least one of them to be installed to run.

5.4.3 OPTIONS

-S <*name*>, --server <*name*>
Specify which WSGI server to use. <*name*> must be one of `gevent`, `gunicorn` or `waitress` and the matching package must then be installed. If the option isn't provided, the first one available will be used.

-h <*hostname*>, --host <*hostname*>
Hostname or IP address on which to listen. The default is `0.0.0.0` which means to listen on all IPv4 interfaces on this host. Cannot be used with **--socket**.

-p <*port*>, --port <*port*>
TCP port on which to listen. Default is `5722`. Cannot be used with **--socket**.

-s <*path*>, --socket <*path*>
Path of a Unix socket on which to bind to. If a path is specified, a Unix domain socket is made instead of the usual inet domain socket. Cannot be used with **--host** or **--port**. Not available on Windows.

--processes <*n*>
Number of worker processes to spawn. Only applicable when using the **Gunicorn** WSGI server.

--threads <*n*>
The number of worker threads for handling requests. Only applicable when using the **Gunicorn** or **Waitress** WSGI server.

5.4.4 BUGS

Bugs can be reported to your distribution's bug tracker or upstream at https://github.com/spl0k/supysonic/issues.

5.4.5 SEE ALSO

`supysonic-cli (1)`

Daemon

5.5 supysonic-daemon

5.5.1 SYNOPSIS

supysonic-daemon

5.5.2 DESCRIPTION

supysonic-daemon is an optional non-exiting process made to be ran in the background to manage background scans, library changes detection and the jukebox mode (audio played on the server hardware).

If **supysonic-daemon** is running when you start a manual scan using **supysonic-cli**, the scan will be run by the daemon process in the background instead of running in the foreground. This daemon also enables the web UI scan feature.

With proper configuration, **supysonic-daemon** also allows authorized users to play audio on the machine's hardware, using their client as a remote control.

5.5.3 BUGS

Bugs can be reported to your distribution's bug tracker or upstream at https://github.com/spl0k/supysonic/issues.

5.5.4 SEE ALSO

`supysonic-cli (1)`

CHAPTER

SIX

SUBSONIC API BREAKDOWN

This page lists all the API methods and their parameters up to the version 1.16.0 (Subsonic 6.1.2). Here you'll find details about which API features Supysonic support, plan on supporting, or won't.

At the moment, the current target API version is 1.12.0.

The following information was gathered by *diff*-ing various snapshots of the Subsonic API page.

6.1 Methods and parameters listing

Statuses explanation:

- : planned
- ✓: done
- : done as not supported
- : won't be implemented
- : not decided yet

The version column specifies the API version which added the related method or parameter. When no version is given, it means the item was introduced prior to or with version 1.8.0.

6.1.1 All methods / pseudo-TOC

Method	Vers.	
ping		✓
getLicense		✓
getMusicFolders		✓
getIndexes		✓
getMusicDirectory		✓
getGenres	1.9.0	✓
getArtists		✓
getArtist		✓
getAlbum		✓
getSong		✓
getVideos		
getVideoInfo	1.15.0	
getArtistInfo	1.11.0	

continues on next page

Table 1 – continued from previous page

Method	Vers.	
getArtistInfo2	1.11.0	
getAlbumInfo	1.14.0	
getAlbumInfo2	1.14.0	
getSimilarSongs	1.11.0	
getSimilarSongs2	1.11.0	
getTopSongs	1.13.0	
getAlbumList		✓
getAlbumList2		✓
getRandomSongs		✓
getSongsByGenre	1.9.0	✓
getNowPlaying		✓
getStarred		✓
getStarred2		✓
search		✓
search2		✓
search3		✓
getPlaylists		✓
getPlaylist		✓
createPlaylist		✓
updatePlaylist		✓
deletePlaylist		✓
stream		✓
download		✓
hls	1.9.0	
getCaptions	1.15.0	
getCoverArt		✓
getLyrics		✓
getAvatar		
star		✓
unstar		✓
setRating		✓
scrobble		✓
getShares		
createShare		
updateShare		
deleteShare		
getPodcasts		
getNewestPodcasts	1.14.0	
refreshPodcasts	1.9.0	
createPodcastChannel	1.9.0	
deletePodcastChannel	1.9.0	
deletePodcastEpisode	1.9.0	
downloadPodcastEpisode	1.9.0	
jukeboxControl		✓
getInternetRadioStations	1.9.0	✓
createInternetRadioStation	1.16.0	✓
updateInternetRadioStation	1.16.0	✓
deleteInternetRadioStation	1.16.0	✓
getChatMessages		✓

continues on next page

Table 1 – continued from previous page

Method	Vers.	
addChatMessage		✓
getUser		✓
getUsers	1.9.0	✓
createUser		✓
updateUser	1.10.2	✓
deleteUser		✓
changePassword		✓
getBookmarks	1.9.0	
createBookmark	1.9.0	
deleteBookmark	1.9.0	
getPlayQueue	1.12.0	
savePlayQueue	1.12.0	
getScanStatus	1.15.0	✓
startScan	1.15.0	✓

6.1.2 Global

Parameters used for any request

P.	Vers.	
`u`		✓
`p`		✓
`t`	1.13.0	
`s`	1.13.0	
`v`		✓
`c`		✓
`f`		✓

Error codes

#	Vers.	
0		✓
10		✓
20		✓
30		✓
40		✓
41	1.15.0	
50		✓
60		✓
70		✓

6.1.3 System

ping

✓

No parameter

getLicense

✓

No parameter

6.1.4 Browsing

getMusicFolders

✓

No parameter

getIndexes

✓

Parameter	Vers.	
musicFolderId		✓
ifModifiedSince		✓

getMusicDirectory

✓

Parameter	Vers.	
id		✓

getGenres

✓ 1.9.0

No parameter

getArtists

✓

Parameter	Vers.	
musicFolderId	1.14.0	✓

getArtist

✓

Parameter	Vers.	
id		✓

getAlbum

✓

Parameter	Vers.	
id		✓

getSong
✓

Parameter	Vers.	
id		✓

getVideos

No parameter

getVideoInfo
1.15.0

Parameter	Vers.	
id	1.15.0	

getArtistInfo
1.11.0

Parameter	Vers.	
id	1.11.0	
count	1.11.0	
includeNotPresent	1.11.0	

getArtistInfo2
1.11.0

Parameter	Vers.	
id	1.11.0	
count	1.11.0	
includeNotPresent	1.11.0	

getAlbumInfo
1.14.0

Parameter	Vers.	
id	1.14.0	

getAlbumInfo2
1.14.0

Parameter	Vers.	
id	1.14.0	

getSimilarSongs
1.11.0

Parameter	Vers.	
id	1.11.0	
count	1.11.0	

getSimilarSongs2
1.11.0

Parameter	Vers.	
id	1.11.0	
count	1.11.0	

getTopSongs
1.13.0

Parameter	Vers.	
artist	1.13.0	
count	1.13.0	

6.1.5 Album/song lists

getAlbumList
✓

Parameter	Vers.	
type		✓
size		✓
offset		✓
fromYear		✓
toYear		✓
genre		✓
musicFolderId	1.12.0	✓

New in version 1.10.1: byYear and byGenre were added to type

getAlbumList2
✓

Parameter	Vers.	
type		✓
size		✓
offset		✓
fromYear		✓
toYear		✓
genre		✓
musicFolderId	1.12.0	✓

New in version 1.10.1: byYear and byGenre were added to type

getRandomSongs
✓

Parameter	Vers.	
size		✓
genre		✓
fromYear		✓
toYear		✓
musicFolderId		✓

getSongsByGenre

✓ 1.9.0

Parameter	Vers.	
genre	1.9.0	✓
count	1.9.0	✓
offset	1.9.0	✓
musicFolderId	1.12.0	✓

getNowPlaying

✓

No parameter

getStarred

✓

Parameter	Vers.	
musicFolderId	1.12.0	✓

getStarred2

✓

Parameter	Vers.	
musicFolderId	1.12.0	✓

6.1.6 Searching

search

✓

Parameter	Vers.	
artist		✓
album		✓
title		✓
any		✓
count		✓
offset		✓
newerThan		✓

search2

✓

Parameter	Vers.	
query		✓
artistCount		✓
artistOffset		✓
albumCount		✓
albumOffset		✓
songCount		✓
songOffset		✓
musicFolderId	1.12.0	✓

search3

✓

Parameter	Vers.	
query		✓
artistCount		✓
artistOffset		✓
albumCount		✓
albumOffset		✓
songCount		✓
songOffset		✓
musicFolderId	1.12.0	✓

6.1.7 Playlists

getPlaylists

✓

Parameter	Vers.	
username		✓

getPlaylist

✓

Parameter	Vers.	
id		✓

createPlaylist

✓

Parameter	Vers.	
playlistId		✓
name		✓
songId		✓

updatePlaylist

✓

Parameter	Vers.	
playlistId		✓
name		✓
comment		✓
public	1.9.0	✓
songIdToAdd		✓
songIndexToRemove		✓

deletePlaylist
✓

Parameter	Vers.	
id		✓

6.1.8 Media retrieval

stream
✓

Parameter	Vers.	
id		✓
maxBitRate		✓
format		✓
timeOffset		
size		
estimateContentLength		✓
converted	1.15.0	

download
✓

Parameter	Vers.	
id		✓

hls
1.9.0

Parameter	Vers.	
id	1.9.0	
bitRate	1.9.0	
audioTrack	1.15.0	

getCaptions
1.15.0

Parameter	Vers.	
id	1.15.0	
format	1.15.0	

getCoverArt

✓

Parameter	Vers.	
id		✓
size		✓

getLyrics

✓

Parameter	Vers.	
artist		✓
title		✓

getAvatar

Parameter	Vers.	
username		

6.1.9 Media annotation

star

✓

Parameter	Vers.	
id		✓
albumId		✓
artistId		✓

unstar

✓

Parameter	Vers.	
id		✓
albumId		✓
artistId		✓

setRating

✓

Parameter	Vers.	
id		✓
rating		✓

scrobble

✓

Parameter	Vers.	
id		✓
time	1.9.0	✓
submission		✓

6.1.10 Sharing

getShares

No parameter

createShare

Parameter	Vers.	
id		
description		
expires		

updateShare

Parameter	Vers.	
id		
description		
expires		

deleteShare

Parameter	Vers.	
id		

6.1.11 Podcast

getPodcasts

Parameter	Vers.	
includeEpisodes	1.9.0	
id	1.9.0	

getNewestPodcasts

1.14.0

Parameter	Vers.	
count	1.14.0	

refreshPodcasts

1.9.0

No parameter

createPodcastChannel
1.9.0

Parameter	Vers.	
url	1.9.0	

deletePodcastChannel
1.9.0

Parameter	Vers.	
id	1.9.0	

deletePodcastEpisode
1.9.0

Parameter	Vers.	
id	1.9.0	

downloadPodcastEpisode
1.9.0

Parameter	Vers.	
id	1.9.0	

6.1.12 Jukebox

jukeboxControl
✓

Parameter	Vers.	
action		✓
index		✓
offset		✓
id		✓
gain		

6.1.13 Internet radio

getInternetRadioStations
1.9.0

No parameter

createInternetRadioStation
1.16.0

Parameter	Vers.	
streamUrl	1.16.0	
name	1.16.0	
homepageUrl	1.16.0	

updateInternetRadioStation
1.16.0

Parameter	Vers.	
id	1.16.0	
streamUrl	1.16.0	
name	1.16.0	
homepageUrl	1.16.0	

deleteInternetRadioStation
1.16.0

Parameter	Vers.	
id	1.16.0	

6.1.14 Chat

getChatMessages
✓

Parameter	Vers.	
since		✓

addChatMessage
✓

Parameter	Vers.	
message		✓

6.1.15 User management

getUser
✓

Parameter	Vers.	
username		✓

getUsers
✓ 1.9.0

No parameter

createUser
✓

Parameter	Vers.	
username		✓
password		✓
email		✓
ldapAuthenticated		
adminRole		✓
settingsRole		
streamRole		
jukeboxRole		✓
downloadRole		
uploadRole		
playlistRole		
coverArtRole		
commentRole		
podcastRole		
shareRole		
videoConversionRole	1.14.0	
musicFolderId	1.12.0	

updateUser

✓ 1.10.2

Parameter	Vers.	
username	1.10.2	✓
password	1.10.2	✓
email	1.10.2	✓
ldapAuthenticated	1.10.2	
adminRole	1.10.2	✓
settingsRole	1.10.2	
streamRole	1.10.2	
jukeboxRole	1.10.2	✓
downloadRole	1.10.2	
uploadRole	1.10.2	
coverArtRole	1.10.2	
commentRole	1.10.2	
podcastRole	1.10.2	
shareRole	1.10.2	
videoConversionRole	1.14.0	
musicFolderId	1.12.0	
maxBitRate	1.13.0	

deleteUser

✓

Parameter	Vers.	
username		✓

changePassword

✓

Parameter	Vers.	
username		✓
password		✓

6.1.16 Bookmarks

getBookmarks
1.9.0

No parameter

createBookmark
1.9.0

Parameter	Vers.	
id	1.9.0	
position	1.9.0	
comment	1.9.0	

deleteBookmark
1.9.0

Parameter	Vers.	
id	1.9.0	

getPlayQueue
1.12.0

No parameter

savePlayQueue
1.12.0

Parameter	Vers.	
id	1.12.0	
current	1.12.0	
position	1.12.0	

6.1.17 Library scanning

getScanStatus
✓ 1.15.0

No parameter

startScan
✓ 1.15.0

No parameter

6.2 Changes by version

6.2.1 Version 1.9.0

Added methods:

- *getGenres*
- *getSongsByGenre*
- *hls*
- *refreshPodcasts*
- *createPodcastChannel*
- *deletePodcastChannel*
- *deletePodcastEpisode*
- *downloadPodcastEpisode*
- *getInternetRadioStations*
- *getUsers*
- *getBookmarks*
- *createBookmark*
- *deleteBookmark*

Added method parameters:

- *updatePlaylist*
  - `public`
- *scrobble*
  - `time`
- *getPodcasts*
  - `includeEpisodes`
  - `id`

6.2.2 Version 1.10.1

Added method parameters:

- *getAlbumList*
  - `fromYear`
  - `toYear`
  - `genre`
- *getAlbumList2*
  - `fromYear`
  - `toYear`
  - `genre`

6.2.3 Version 1.10.2

Added methods:

- *updateUser*

6.2.4 Version 1.11.0

Added methods:

- *getArtistInfo*
- *getArtistInfo2*
- *getSimilarSongs*
- *getSimilarSongs2*

6.2.5 Version 1.12.0

Added methods:

- *getPlayQueue*
- *savePlayQueue*

Added method parameters:

- *getAlbumList*
  - `musicFolderId`
- *getAlbumList2*
  - `musicFolderId`
- *getSongsByGenre*
  - `musicFolderId`
- *getStarred*
  - `musicFolderId`
- *getStarred2*
  - `musicFolderId`
- *search2*
  - `musicFolderId`
- *search3*
  - `musicFolderId`
- *createUser*
  - `musicFolderId`
- *updateUser*
  - `musicFolderId`

6.2.6 Version 1.13.0

Added global parameters:

- `t`
- `s`

Added methods:

- *getTopSongs*

Added method parameters:

- *updateUser*
  - `maxBitRate`

6.2.7 Version 1.14.0

Added methods:

- *getAlbumInfo*
- *getAlbumInfo2*
- *getNewestPodcasts*

Added method parameters:

- *getArtists*
  - `musicFolderId`
- *createUser*
  - `videoConversionRole`
- *updateUser*
  - `videoConversionRole`

6.2.8 Version 1.15.0

Added error code `41`

Added methods:

- *getVideoInfo*
- *getCaptions*
- *getScanStatus*
- *startScan*

Added method parameters:

- *stream*
  - `converted`
- *hls*
  - `audioTrack`

6.2.9 Version 1.16.0

Added methods:

- *createInternetRadioStation*
- *updateInternetRadioStation*
- *deleteInternetRadioStation*