# Squeeze2upnp User Guide

## 1 Introduction

### 1.1 What Is It?

This program (squeeze2upnp - sq2u for short) allows UPnP/DLNA renderers to be used and controlled from LMS as regular squeezebox devices. It acts as a proxy between LMS and UPnP/DLNA renderers. It scans the local network for renderers and creates an instance of an LMS software player for each renderer found.

It can run on any computer running Windows or Linux. For speed reasons, it is better to be co-located with LMS, but this is not essential.

### 1.2 How it Works

Sq2u receives commands and audio streams from LMS. All commands sent by LMS are transformed into UPnP commands. Timing (position) information is taken every second from the UPnP renderer so the LMS cursor might go back and forth a bit as LMS first tries to guess the position and then "match" it with what it receives.

Audio streams are buffered and forwarded to the UPnP renderer, without changing the audio coding format. For buffering it creates two local files, which can be as large as the original files sent by LMS (there are ways of limiting the size of these files if required).

Gapless play is attempted by using a UPnP feature that allows data to be sent for the next track whilst playing the current one. This is not available on all renderers.

Reply Gain is not available, nor Smart Transition (fading in and out) or multi-player synchronisation.

## 2 Quick Guide

### 2.1 Installation

Copy the sq2u program to wherever you wish for your system and platform, optimally on the same machine as LMS. There is no installation process (there is no installation 'package').

On Windows, you need to copy pthreadBC2 and cc32160mt.dll to the same directory as the program.

On Linux (Intel CPU), if you use a 32 bit system, just copy squeeze2upnp-x86. If your system is 64bits, you need to have libc6 32 bits. To install it, do, at any command prompt:

*sudo apt-get install libc6:i386*

### 2.2 Identify Your UPnP/DLNA Renderers

Follow these steps:

i. Turn on all the renderers you might ever want to use and make sure they are connected to your local network.
ii. Run the squeeze2upnp program from a command line. Wait for about 1 minute.
iii. Type save config.xml
iv. Type exit

-or-

ii. Run the “squeeze2upnp –i config.xml” from a command line
iii. After less than 1 minute, the program exits

In the directory containing sq2u you should now have a file called config.xml. This lists all the renderers found by sq2u and contains the default configuration values. Take a copy of this file and save it somewhere. It may come in useful some day.

If you ever add a new renderer to your system you will need to repeat this process. If you 'save' again to your current configuration file your existing options will be carried over (but currently un-detected players will be removed).

## 2.3 Adjust Configuration

With a bit of luck all your renderers will be 'standard', or what passes for standard in the UPnP world. If not, you will need to read more of this manual.

The configuration file you now have (config.xml) works with many systems. It consists of a general header, some common parameters, and then a section for each and every renderer sq2u has found.

By default sq2u only supports mp3 streams. If your LMS does not have access to the LAME program you may find that files or streams in formats other than mp3 will not work. One simple change you can try in the configuration file is to change:

```
<codecs>mp3</codecs>
```

to

```
<codecs>flc,mp3</codecs>
```

sq2u will then support flac format files and streams as well, on all your renderers.

If some of your renderers support different types of format you can tailor the codecs for each renderer. See section 3.5 for further details. You may also need to refer to the some format settings on LMS. Using the LMS GUI in a web browser, go to:

- Settings
- Advanced Tab
- In the drop down select File Types

Look at the various formats (aac, aiff, flac, mp3 and so on). Each format on the left has a number of options on the right hand side, some of which may be disabled. For each source format you use, make sure that there is a format supported by your renderers on the right hand side. For the current version of sq2u this must be any or all of flac, mp3 and pcm (aac may also work on some systems).

## 2.4 Start The Program

If you are happy for all files related to sq2u to be in the same directory as the program, and your user has write access to it, just run the program in a terminal. Within a short time the renderers will appear as renderers in the LMS GUI and you should be able to play tracks as normal with any squeezebox device (or squeezelite, squeezeplay etc.).

**It is strongly suggested that before you start to play on a device you check the volume for the device in LMS is not near the maximum, so as to avoid damaging your speakers. When using**

**sq2u and LMS to control a renderer you should avoid using any controls on the renderer itself (or its remote), except the volume control in some cases (there is an option to allow this), since LMS will not recognise such changes and may override them. If you find that when you start up your renderer always plays the last song it played, possibly at high volume, either make sure you clear the LMS playlist before you stop, or better still set the LMS player's parameter under "settings", “player”, "audio", "power on resume" to "stop at power off, remain stopped at power on”.**

If you would rather not have the log being output to the terminal, you can specify a file for the log using the '-f' option. Run the program with the '-h' option to see what other command line options are available.

Type exit to stop the program.

If you want to automate starting the program see Section 3.3.

# 3 Full Guide

## 3.1 Install or Compile

You can copy the squeeze2upnp program to the location where you want to run it, or compile it yourself.

If you are familiar with compiling programs on your platform, go to https://github.com/philippe44/LMS-to-uPnP/tree/master/stable where the stable version of the source of the program is located. The source can be downloaded and built in the normal way. The README contains compilation instructions.

Note that you need to apply patches to version 1.6.19 of libupnp, and build it. You then need to ensure the Makefile can find the include files and libraries in libupnp.

## 3.2 Startup Options

To see the command line switches, run squeeze2upnp with -h. Some of these options, such as the server id, can be overridden in the configuration file.

You have a choice as to where to place logs and where the configuration file is located. The configuration file also contains the location of the audio data files created and used for buffering by sq2u. Make sure the log and data files are in directories that sq2u can write to.

On startup, sq2u tries to open a file named “config.xml” in the current directory (or uses the filename provided using the –x option). If no file is found, all parameters will be set to default. To generate such a configuration file, type “save <filename>” when sq2u is running and a file with all existing configuration parameters and discovered devices will be created.

If you are knowledgeable about such things, you can arrange to start sq2u at boot time as a 'service' on your platform. If running on linux you can 'daemonize' the program (i.e. make it run as a background service) when you start it.

## 3.3 Using Server Power Control to Start Squeezeupnp

If running sq2u on a headless system (i.e. one with no normal access to keyboard or monitor) you may want to be able to start and stop the program remotely. If you have your own way of doing this, fine. If not, you can, if you wish, use a method from within the LMS GUI.

There is an LMS plugin called Server Power Control. Go to the Settings, Plugins page. In Additional Repositories at the bottom, add:

> http://srvrpowerctrl.googlecode.com/svn/beta.xml

And 'Apply' the change. Now find and enable the Server Power Control (Beta) plugin (under Gordon Harris's plugins), 'Apply' the change and restart LMS.

Then go to Settings, Advanced and in the drop-down find Server Power Control.

Add a Custom Command to start your sq2p program with whatever parameters you need. It may be easier to create a batch or shell command file containing the commands to stop any existing copy of the program and then start the program. A linux example might look like:

```
#!/bin/bash
pkill -SIGKILL -f squeeze2upnp
/home/user/Programs/squeeze2upnp/squeeze2upnp-x86 -z
```

```
-x/home/user/Programs/squeeze2upnp/sq2upnpconfig.xml
-f /home/user/Downloads/squeeze2upnp.log -d all=info
```

In the Server Power Control custom command then enter a name for the command, and the full path to the command file.

If you create another entry with the name '.autoexec' to execute the same command, then when LMS starts up, sq2u will also start up about 30 seconds later.

You might also find it useful to have a 'stop' command file as well.

'Apply' the changes (bottom right). The command should now appear on the LMS GUI Extras list on the Home page under 'Server Power Control'.

Finally take another look at the Server Power Control settings and make sure you disable anything you don't want (e.g. powering down LMS after a period of inactivity).

## 3.4 Runtime Options and Actions

There a few commands that can be entered on the command line while sq2u is running:

- exit : terminates
- save <filename> : save the config file
- stop * or stop <string> will stop any renderer whose name matches <string>

Searches for new (or enabled but not yet running) UPnP devices will occur every 2 minutes and last for 15 seconds. The first search will last 15 seconds before LMS devices are created.

If a renderer disappears from the LAN, it will be automatically removed after a maximum of 2 minutes.

## 3.5 Renderer Variations

UPnP devices being non-standard animals, there are various oddities with the translation of the commands, tuned by parameters described in a XML configuration file described in Section 4.1.

Go to https://github.com/philippe44/LMS-to-uPnP/tree/master/stable and look at 'compatibility'. This lists several renderers and the options that have been found to work for them.

The main variations are in the codecs (audio formats) supported, sample size, sample rates and gapless play. It may be possible to find out more about your renderer from your user manual. If not, you may be able to use a software tool to find out more.

Device Spy (Windows) and UPnP Inspector (Linux) are two such tools. The main thing you are interested in is in the 'Connection Manager', 'GetProtocolInfo', which supplies 'ProtocolInfo' for the renderer. This should list the codecs, sample size (16, 24 or 32 bits) and sample rates (44100, 48000, 88200, 96000 Hz, etc.). For the codecs and sample rates, set the options for the specific renderer in the <device> section of the configuration file. LMS should then transcode any other codecs or higher sample rates to those your system supports.

If no supporting codec can be found for a track sq2p will write an error message to the log and skip to the next track (*in version 0.1.4 and above*).

If your renderer supports wav but not pcm, sq2u will add a wav header to the pcm stream from LMS.

The sample size is a little more difficult. If you have 24 or 32 bit sources and your system only supports 16 bits, you should use a codec such as flc (flac) so that LMS can resample the audio data. If you have 24 bit sources and use the pcm codec when your system only support 16 bit pcm it will not work. Renderers that support 24 bit pcm packed type 1 need to have <L24_format> set to 1

All file formats, except pcm, include a header or information inside the stream itself so that the decoder knows the essential information (sample size, rate and channels). However, pcm is 'raw' and so the renderer needs this information to be sent in the HTTP response header otherwise it has no idea of what it will receive. This is the MIME-TYPE, like "audio/L16;channels=2;rate=44100" for pcm 16 bits, 2 channels 44100Hz. For wav, MIME-TYPE would be audio/wav and for flac audio/flac (nothing else is needed). Normally, in the ProtocolInfo section of the player it states all this detail for all items it supports (audio/L16;channels=2;rate=44100, audio/L16;channels=2;rate=48000, audio/flac and so on). But some renderers just say "audio/L16" so sq2u has no idea what they really support. In that case sq2u sets the MIME-TYPE to be audio/L16:channel=x;rate=y where x and y are channel and sample rate according to what LMS has requested, but this may not work.

To determine if your renderer can support gapless playing, Device Spy or UPnP Inspector can show this under the AVTransport section. You should see both SetAVTransportURI and SetNextAvTransportURI. If the the second one is missing, your player does not support gapless playing and you will need to set <accept_nexturi> in the configuration file to 0.

One of the other areas where renderers behaviour varies is on support for pausing mid-track, and for re-positioning a track part way though playing. The Configuration file has several options relating to these facilities, More detail can be found in Section 3.8. Some renderers just have problems re-positioning when playing flac files. See Section 3.6 au-dessous on how to fix this.

If tracks continue while they are supposed to be paused, set <seek_after_pause> to 1 (sound will always be muted during a pause and LMS will look paused, but it will be clear if the track has moved on when you un-pause).

## 3.6 LMS Options

LMS has many options. The main ones that may affect sq2u are the LMS File Types (see Settings, Advanced, in the Drop Down). This is discussed in Section 2.3 above.

Some renderers do not support re-positioning of flac well. If you have this problem you need to create a custom transcoding rule. You first need to find the convert.conf file on your system (on linux this is in /etc/squeezeboxserver). This file contains the rules LMS uses to transcode from one format to another. To add a new rule, create a new file called 'custom-convert.conf' in the same directory and restart LMS. This file should contain:

*(if you copy-paste below,be careful with the indentation)*

```
flc flc * *
    # IR
     [flac] -cs - -
```

This tells LMS to create a complete new flac file when a flac starts playing or is re-positioned. The LMS File Type for flac should now show flac transcoded to 'flac' rather than 'native'.

If you do not have LAME installed you may have problems with other codecs being transcoded to mp3s. If you cannot install LAME, make sure all your other required source codecs translate to flac or pcm in the LMS File Types (mp3s should still play natively).

Note that sq2u uses the LMS Command Line Interface (CLI) , and assumes port 9090. This is needed for the <seek_after_pause> option to work (see Section 4.1) and to send metadata like artist, album, and title to your renderer. CLI login and password is currently not supported, so make sure LMS is configured to work without this. There is a <send_metadata> option *in version 0.1.4* that can be unset to suppress the metadata, so if this and <seek_after_pause> are suppressed, the login is not required.

It is also worth looking at the LMS settings for each configured player. See LMS, Settings, Players for more details. You may, for instance, want to ensure a renderer is fully stopped when LMS stops and does not restart when LMS starts up.

## 3.7 Interoperability with UPnP renderer dedicated remote

LMS takes full control of the renderer. In the squeezebox system, Infra Red or other remote control software send LMS commands that are then forwarded by LMS to real squeezebox players.

In the case of a UPnP renderer "proxied" (registered) by sq2u, you want to use the renderer's dedicated remote, use commands directly on the renderer, use other UPnP control software or simply use the renderer independently from LMS without stopping sq2u.

To use a registered renderer independently from LMS, it is best is to switch it off on LMS, using the GUI. Then whatever happens to the renderer is not forwarded to LMS and no LMS command will be sent that would interfere with it. Only when you switch the renderer back on from LMS will it be under the control of LMS.

Having to switch a UPnP renderer off every time you want to use it independently from LMS might be inconvenient. Unfortunately, LMS can send volume commands even to a "stopped" renderer and that can be inconvenient. To avoid these "spurious" volume commands (unless you use LMS to send an explicit "play" request), set the option <volume_on_play> to 1.

Now, what if you want to use the renderer with LMS, but want to use some of the renderer's own controls at the same time? Doing this is not recommended, but starting with version 0.1.4.0, pause and play commands requested using the renderer's own controls are forwarded to LMS so that it synchronizes accordingly. This takes up to 1 second to be taken into account, so a frenetic series of play/pause commands will create problems. This facility may not work well with all renderers.

Volume commands are not currently synchronized, so any change directly on the renderer will be overridden by LMS at next track or at the first pause/play sequence. If you want to only control volume from the renderer itself and ignore any LMS volume commands, set <volume_on_play> to -1 and LMS will have no influence on the renderer's volume.

Using "next" or "previous on the renderer directly does not work, and never will.

Using other UPnP software to control the same live stream as LMS/squeeze2upnp is not recommended.

## 3.8 Troubleshooting <stream-length> and <seek_after_pause>

One common issue is that renderers are correctly detected but when you press "play", nothing happens or even worse, the renderer crashes.

Always start with <codecs> set to mp3 only, use an mp3 file (or have LMS to transcode to mp3) and use tracks encoded with parameters such as 16 bits and 44.1KHz.

Unlike most UPnP controllers LMS does not send track size information to its players and this creates a major difficulty for sq2u because many UPnP renderers expect it. The track transfer between sq2u and UPnP renderers uses HTTP, and the parameter <stream-length> allows using variations of that protocol to work around potential problems.

Start with -3, and if that does not work try -2. If it still does not work, try -4 and then -1. If none of these work set it to a large positive value like 100000000. This value is your estimation of the largest file that LMS will send (which must be below 2^32 bytes). A detailed explanation of these modes can be found below.

Once the renderer plays properly, test that "pause" and "un-pause" work. First try with <seek_after_pause> set to 0. If you observe that the renderer has not stopped (only muted, but the track continued) during a "pause" or if the renderer restarts from the beginning of the current track, then, change the <stream_length> according to the test sequence above. If no value allows pause to work, then restore your initial <stream_length> value and set <seek_after_pause> to 1.

### 3.8.1 Technical Explanation

HTTP allows clients to request a specific range of bytes from the server, rather than the complete file, and that range can be "open-ended" (i.e. start from byte N, but no end of range is given). Some complications arise when the server does not know the size of the file to be sent AND the client asks for an open-ended range. Typically some clients always send a request for 1st byte to "no-end", which requires a HTTP PARTIAL REPONSE, instead of a sending a simple request with no range (whole file) that would have required a HTTP OK response.

<stream_length> of -3 means use "chunked encoding" which is designed for HTTP transfers when the size of the source is unknown (e.g. radio streams). The server sends data in "chunks" and indicates when the last "chunk" has been sent, so the client does not need to know the size in advance. However, the HTTP standard is unclear about open-ended range-requests and chunked-encoding. With -3, sq2u answers with a HTTP PARTIAL REPONSE formatted accordingly to "common practices" (no content-length, content-range: 0-*). However, it seems that some renderers don't know how to handle this response.

<stream_length> of -2 is almost identical to -3 (uses chunked-encoding) but open-ended requests are responded to with an HTTP OK response (no content-length). According to the HTTP standard, it is accepted but not recommended.

<stream_length> set to a positive value is used to "cheat" and tell the UPnP renderer that the sent file length is of known (content-length). This disables chunked-encoding and sq2u will send data until it has no more, then it will close the connection. Some UPnP renderers might see that as an error as not all the "promised" data have been received. Some others might want to seek from the end of the file and that causes grief to sq2u. One of the benefits of "faking" a file size is that all range-requests will be properly responded to.

<stream-length> of -4 is similar to setting a positive value, but no file size information is sent to the UPnP renderer (no content-length). In response to an open-ended byte range request, only up to 1MB of data is returned, following the recommended HTTP PARTIAL REPONSE standard for unknown size (if N is the number of bytes, then content-length: N and content-range: 0-(N-1)/*). The UPnP renderer shall then query the rest of the file.

<stream-length> of -1 is similar to -4, but open-ended requests are responded with an HTTP OK response (no content-length) and the full file is sent. According to the HTTP standard, it is accepted but not recommended.

When some UPnP renderers receive files with an unknown size they treat them as live streams that cannot be paused, even though sq2u has indicated that it can pause and retain a complete stream (only valid for DLNA renderers). In that case, try to use a <stream_length> with a real value or as a last resort, use <seek_after_pause> at 1.

With <seek_after_pause> set to 1, when LMS asks a player to “un-pause”, sq2u will request the current position and then issue a “reposition” command to LMS before sending a “play” request to the UPnP renderer. A side effect is a 1 to 2 second time difference between the “paused” and the “un-paused” positions.

# 4 Configuring Squeeze2uPnP

## 4.1 The Configuration File

The full set of options is as follows (comments are shown in italics and do not appear in the file). A number of parameters are there for future use only, and commented N/A (Not Applicable).

**<?xml version="1.0"?>**
**<squeeze2upnp>**

*Global options*
**<server/>[ip:port | ?]</server>** *default = ?*
*ip = IP address or network name, port = port number (usually 3483) of your LMS server.*
*Use ? for auto-discover*
*This can be overridden in the command line with the -s option.*

**<slimproto_stream_port>[port]</slimproto_stream_port>**
*N/A.*

**<base_mac>[xx:xx:xx:xx:xx:xx]</base_mac>** *default = 00:01:02:03:04:05*
*By default, the MAC of the UPnP renderer is used to identify it. If sq2u cannot read this MAC address (for instance from software renderers) then the first such renderer discovered will start with this <base_mac>, the 2nd with <base_mac> + 1 and so on.*

**<upnp_scan_interval>[0 | value]</upnp_scan_interval>** *default = 120*
*Sets how often (in seconds) a network scan is done to update upnp media renders list (and add or remove corresponding LMS players). 0 disables rescan (only one scan upon startup)*

**<slimproto_log>[error | warn | info | debug | sdebug]</slimproto_log>**
**<stream_log>[error | warn | info | debug | sdebug]</stream_log>**
**<output_log>[error | warn | info | debug | sdebug]</output_log>**
**<decode_log>[error | warn | info | debug | sdebug]</decode_log>**
**<web_log>[error | warn | info | debug | sdebug]</web_log>**
**<upnp_log>[error | warn | info | debug | sdebug]</upnp_log>**
**<main_log>[error | warn | info | debug | sdebug]</main_log>**
**<sq2mr_log>[error | warn | info | debug | sdebug]</sq2mr_log>**
*Level of debugging for each sub-item of sq2upnp. Leave these at info in the current version.*
*These can be overridden in the command line with the -d option.*

**<common>**
*All items in this sub-section set the default behaviour for all renderers. Any option can be repeated in the <device> section of a given renderer to override the <common> option.*

**<enabled>[0 | 1]</enabled>** *default = 1*
*'1' to automatically add any discovered UPnP renderer to LMS players, even if there is no <device> section referring to it. '0' to prevent this.*

**<streambuf_size>[value > 1000000]</streambuf_size>** *default = 2457600*
*Size of the internal memory buffer used to store audio data sent by LMS. Any value above 1,000,000 is fine, 2,457,600 is recommended. If your network is slow you may need to increase it.*

**<output_size>[value > 2000000]</output_size>**
*N/A*

<**buffer_dir>[path with /]</output_size>** *default = .*

*The Directory where the stream data are buffered. Use '.' for the current directory. Two files are created for each renderer : xx-xx-xx-xx-xx-xx-idx-0 and xx-xx-xx-xx-xx-xx-idx-1 where xx-xx-xx-xx-xx-xx is the MAC address of the renderer. These files can be as big as the largest file played by LMS (but see also <buffer_limit> below). In case of live streams, these file grow constantly, so be careful. Either press "next" on the live stream from time to time to create a new file, or set the <buffer_limit> below.*

**<buffer_limit>[-1 | value]</buffer_limit>** *default = -1*
*The maximum size of each stream data buffer. When a buffer reaches this limit is it shrunk by half. As a consequence, sq2upnp "loses" memory of data received before the shrinkage happens and cannot seek back to any of "lost" portion. In some cases, this might prevent "pause" from working properly after the buffer has been shrunk once, especially for renderers that cannot support <stream_length> of -3 (e.g. Sonos). Note that if a renderer is "paused" on a live stream for an undefined amount of time, the buffering of the stream will be left to LMS and as a result audio loss might happen. Set this to around 32MB for normal running. If this causes any problems (such as sound cut-outs) with very high quality audio and you have a load of spare space then set this very high indeed to avoid the buffer ever being shrunk on local files (e.g. 2GB, not more than 2^32-1). Alternatively set this parameter to -1 to let the buffer grow without a limit. The files will be replaced when playing stops or sq2p is restarted from cold..*

**<stream_length>[-1 | -2 | -3 | -4 | value > 100000000]</stream_length>** *default = -3*
*-1 : infinite transfer, using HTTP200 for open-ended range request*
*-2 : chunked transfers, using HTTP200 for open-ended range request*
*-3 : chunked transfers (recommended), using HTTP206 for range request*
*-4 : transfer up to close, using HTTP206 for open-ended range request*
*value : any large number*
*This tells sq2u how to transfer the data in HTTP transfers. LMS does not state in advance the length (either in bytes or seconds) of the audio stream that it will send. Normally, HTTP transfers should use -3 which allow data to be transferred in "chunks" until there is nothing to transfer. When there is nothing left, sq2u sends an empty chunk so that the end of stream will be assumed. However, some renderers do not seem to like "chunked" mode so try a value larger than the largest of your audio encoded tracks - try between 100,000,000 and 8,000,000,000.For a detailed explanation of these values see Sections* 3.8 *and* 3.8.1.

**<max_read_wait>[value > 10]</max_read_wait>** *default = 100*
*When the buffer of data from LMS is empty but more data is expected from LMS, this option sets the amount of time in Nx50ms (where value = N) that sq2u waits before signalling end of stream to the renderer. Recommended value is 100 (a 5 second wait).*

**<max_GET_bytes>[0 | -1 | value > 100000]</max_GET_bytes>** *default = -1*
*Limit the size of a chunk of stream data sent by sq2u to a renderer. This helps slowing down internal sq2u buffer consumption and preventing overrun, especially with slow networks.*
*0 = value "recommended" by LMS*
*-1 = Max sq2u can send (1MB) – recommended*
*value > 100,000*

**<process_mode>[0 … 3]</process_mode>** *default = 2*
*The mode of operation – leave at 2 for this version of sq2p*

**<seek_after_pause>[0 | 1]</seek_after_pause>** *default = 0*
*Sq2u allows renderers to leave open and "stall" the TCP connection as long as they want for pausing. However, some renderers will chose to close the connection and to re-open-it at un-pause. Normally, renderers will then either request sq2u to seek to the last received position or will re-start from 1st byte but consume all the intervening data. Other renderers either refuse to pause and will let the track move on, or will not un-pause from the correct position. In such cases, set this parameter to 1 and sq2u will issue a "reposition" request to LMS. This uses the CLI interface to instruct LMS to cue to the paused position, so if you have a login/password set, it will fail as there is no option today to have sq2u to use a login/password.*

**<force_volume>[0 | 1]</force_volume>** *default = 0*

*Some renderers do not accept the ramp-up/down volume commands sent by LMS at each stop/pause/start and will miss some of these volume commands. Set to 1 this parameter forces sq2u to re-issue a "set volume" command after the renderer has confirmed that a track has started to play.*

<**volume_on_play**>**[-1 | 0 | 1]**</**volume_on_play**> *default = 1*
*LMS sends a volume control command (set to 0) whenever a new renderer is registered (at startup or any other time). This can be an annoying and unwanted effect when such renderer is not actively controlled by LMS (although it is registered). One way to avoid this is to set the renderer "off" in LMS. However, you may want to use the renderer for other purposes without having to switch it "off" in LMS. With this parameter set to 1 sq2u will not obey any volume command from LMS until it is requested to "play" a track (volume command are still memorized, though). If you want to ignore all volume commands sent by LMS and only use the renderer's own volume control, set this to -1. Note also that when "off" or when stopped, sq2u will ignore all UPnP events to avoid unsolicited reaction from LMS when a UPnP renderer is being controlled by some other mean than LMS.*

**<volume_curve> [xx:yy, xx:yy, …] </volume_curve>**
*default = 0:0, 400:10, 700:20, 1200:30, 2050:40, 3800:50, 6600:60, 12000:70, 21000:80, 37000:90, 65536:100*
*LMS sends volume values from 0 to 65536, although on the user interface it shows as 0 to 100%. The translation between the two values is log, not linear and is renderer dependent (i.e. 20% is 700, 50% = 3800 and 80% is 21000). This parameter re-linearizes the translation. This is a comma-separated list of pair [send_value:percentage]. Any LMS request between two points is a linear approximation between these. Change it to match your taste of volume control.* ***NOTE:- Some renderers seem to output maximum volume at levels below 100%, and so some users have set the upper limit to, say 30, or even 15. If you have any doubts about this it may be worth setting a low upper limit initially and then increasing it as needed, so as to avoid any damage to your speakers.***

**<accept_nexturi>[0 | 1]</ accept_nexturi >** *default = 1*
*Gapless play is by setting a "current track/URI" and a "next track/URI" in UPnP renderers. When they reach the end of "current track/URI", renderers move to the next one automatically. Some renderers do not support this feature and will stop when the "current track/URI" ends. This options forces the next track to be played on faulty renderers, but there will be a gap. Some renderers buffer the "next track/URI" while playing the "currentURI" to do real gapless. Other renderers, although they support "next track/URI", wait for the end of the current track before buffering the next one, and that will create a gap.*

**<codecs>[codec1, codec2, …]</codecs>** *default = mp3*
*The comma separated list of codecs, supported by the UPnP renderer, that will be reported to LMS. If a track is encoded in a non-supported format, LMS will apply its transcoding rules. The codecs are treated in priority preference order, so if, say, pcm is first, it will be the chosen codec if the source can be made available in that codec. pcm, flc and mp3 are supported by sq2u (and aac may also work on some systems). When LMS pauses it locally generates a new "truncated" file and sends a new command to play/stream that file. Unfortunately, it does not seem to insert a header on the truncated file. Some UPnP renderers do not like that, so a header is "guessed" and re-inserted by sq2u. This is only done for flac. Note that sq2u itself does no transcoding from one format to another. See also Section* 3.5.

**<sample_rate>[32000 | 44100 | 48000 | 88200 | 96000 | 176400 | 192000]</sample_rate>** *default = 48000*
*This is the maximum sample rate accepted by the UPnP renderer that will be reported to LMS. If a track is encoded at a higher rate, LMS will apply resampling rules. Note that sq2u itself does no resampling.*

**<L24_format>[0 | 1]</L24_format>** *default = 1*
Defines what kind of 24 bits sample the renderer can handle.
*0: packed format with 3 bytes only*
*1: packed format according to LPCM type 1 packing (similar to the DVD LPCM format)*

**<flac_header>[0 | 1 | 2]</flac_header>** *default = 1*
*When repositioning flac streams, LMS re-generates a file starting from the new position but does not insert a proper header . Such header is required by some players and it contains information like total number of*

*samples in the file and a checksum of the decoded audio data. Sq2u cannot know these values but the flac standard specifies that they should then be set to 0. In a few cases, some players consider such audio data as being a live stream and will consequently refuse to pause. The symptoms varies, but mainly after a re-positioning, pause/un-pause will fail or, if <seek_after_pause> is set to 1, the first pause/un-pause will work, but not any subsequent pause/un-pause. A workaround is to set these header value to a fixed but large value. For other players, the simple solution is to not add a header.*
*0 : no header re-inserted*
*1: header re-inserted with 0 for total number of sample and checksum*
*2: header re-inserted with 2^32-1 for total number of samples and 0xaa (170) 16 times for checksum*

**<send_metadata>[0 | 1]</send_metadata>** *default = 1*
*If you do not want to see metadata (Artist, Album name etc.) on your renderer, or if you have a login and password for the LMS CLI interface, set this to 0. Version 0.1.4.0 on.*

**</common>**

*This is the beginning of the sections that describes each device. There must be one <device> section for each UPnP renderer. Any parameter set in the <common> can be repeated here, to override the defaults.*

*first device …*

**<device>**

<**udn>uuid:***RINCON_000E58FD8A4A01400***</udn>**
*!!! DO NOT EDIT THIS !!! It is the UPnP device unique identifier for that renderer – this is discovered automatically. The example is is for a Sonos renderer.*

**<name>[string]</name>**
*The name of the renderer as seen in LMS – can be set to whatever you want. Defaults to the "UPnP friendly" name.*

**<mac>[xx:xx:xx:xx:xx:xx]</name>**
*The mac address of the player. If this parameter is omitted, sq2u will try to discover the true physical address. If it cannot be discovered, then <base_mac> + N will be used, where N is the number of players already discovered. Once this parameter is set, true physical discovery is not used. <u>All players must have unique mac for LMS to work properly</u>*

**<enabled>[0 | 1]</enabled>** *default = 1*
*Set to 1 to allow the UPnP renderer to be managed be sq2u. An LMS device will be created for each renderer whose mode is above 0. Set to 0 to ignore the renderer. It may be better to keep a list of all known renderers in the configuration and disable those you don't want to use, so that you have a list of the uuids should you ever need them.*

**</device>**

*next device …*
**<device>**
*…*
**</device>**

**</squeeze2upnp>**

## 4.2 Example Configuration File

```
<?xml version="1.0"?>
<squeeze2upnp>

<server>?</server>
<slimproto_stream_port>9000</slimproto_stream_port>
<base_mac>00:01:02:03:04:05</base_mac>
<slimproto_log>info</slimproto_log>
<stream_log>info</stream_log>
<output_log>info</output_log>
<decode_log>info</decode_log>
<web_log>info</web_log>
<upnp_log>info</upnp_log>
<main_log>info</main_log>
<sq2mr_log>info</sq2mr_log>

<common>
        <streambuf_size>2457600</streambuf_size>
        <output_size>2457600</output_size>
        <buffer_dir>.</buffer_dir>
        <buffer_limit>-1</buffer_limit>
        <stream_length>-3</stream_length>
        <max_read_wait>100</max_read_wait>
        <max_GET_bytes>-1</max_GET_bytes>
        <enabled>1<enabled>
        <process_mode>2</process_mode>
        <codecs>mp3</codecs>
        <sample_rate>48000</sample_rate>
        <L24_format>1</L24_format>
        <flac_header>1</flac_header>
        <seek_after_pause>0</seek_after_pause>
        <force_volume>1</force_volume>
        <volume_on_play>1</volume_on_play>
        <volume_curve>0:0, 400:10, 700:20, 1200:30, 2050:40, 3800:50, 6600:60, 12000:70, 21000:80, 37000:90,
                65536:100</volume_curve>
        <accept_nexturi>1</accept_nexturi>
        <send_metadata>1</send_metadata>
</common>

<device>
        <udn>uuid:RINCON_000E58C202B601400</udn>
        <name>192.168.2.24 – Sonos PLAY:1</name>
        <enabled>0</enabled>
</device>

<device>
        <udn>uuid:RINCON_000E58FD8A4A01400</udn>
        <name>192.168.2.23 - Sonos PLAY:3</name>
        <enabled>1</enabled>
        <codecs>flc,mp3</codecs>
</device>

<device>
        <udn>uuid:RINCON_000E58C3F67C01400</udn>
        <name>192.168.2.22 - Sonos PLAY:1</name>
        <enabled>1</enabled>
        <codecs>flc,mp3</codecs>
</device>
</squeeze2upnp>
```