pa-dlna

Release 1.2.1

Jun 17, 2026

TABLE OF CONTENTS

1 pa-dlna 1.2.1 **1**
- 1.1 Requirements . . . 1
- 1.2 Installation . . . 2
- 1.3 Configuration . . . 3

2 Usage **5**
- 2.1 *pa-dlna command* usage . . . 5
- 2.2 *upnp-cmd command* usage . . . 7
- 2.3 UPnP Library . . . 7

3 Configuration **9**
- 3.1 Encoder selection . . . 9
- 3.2 Streaming . . . 9
- 3.3 Encoders configuration . . . 10
- 3.4 User configuration . . . 10
- 3.5 PulseAudio options . . . 11
- 3.6 Common options . . . 12
- 3.7 Encoder specific options . . . 12

4 Built-in Default Configuration **13**

5 upnp-cmd command **17**
- 5.1 Synopsis . . . 17
- 5.2 Options . . . 17

6 pa-dlna command **19**
- 6.1 Synopsis . . . 19
- 6.2 Options . . . 19

7 systemd **21**
- 7.1 Usage . . . 21
- 7.2 The pa-dlna.service unit . . . 21

8 Development **23**
- 8.1 Design . . . 23
- 8.2 Development process . . . 25

9 Release history **29**

Index **33**

CHAPTER

ONE

PA-DLNA 1.2.1

pa-dlna forwards audio streams to DLNA devices.

A Python project based on asyncio, that uses ctypes to interface with the `libpulse` library and supports the PulseAudio and PipeWire[1] sound servers.

pa-dlna is composed of the following components:

- The `pa-dlna` program forwards PulseAudio streams to DLNA devices.
- The `upnp-cmd` is an interactive command line tool for introspection and control of UPnP devices[2].
- The UPnP Python sub-package is used by both commands.

The documentation is hosted at Read the Docs:

- The stable documentation of the last released version.
- The latest documentation of the current GitLab development version.

To access the documentation as a pdf document one must click on the icon at the down-right corner of any page. It allows to switch between stable and latest versions and to select the corresponding pdf document.

1.1 Requirements

Python version 3.8 or more recent.

1.1.1 psutil

The UPnP sub-package and therefore the `upnp-cmd` and `pa-dlna` commands depend on the psutil Python package. This package is available in most distributions as `python3-psutil` or `python-psutil`. It will be installed by `pip` as a dependency of `pa-dlna` if not already installed as a package of the distribution.

1.1.2 libpulse

libpulse is a Python asyncio interface to the Pulseaudio and Pipewire `libpulse` library. It was a sub-package of `pa-dlna` and has become a full-fledged package on PyPi. It will be installed by `pip` as a dependency of `pa-dlna`.

[1] When using PipeWire with the Wireplumber session manager, `pa-dlna` must be started before the audio streams that are routed to DLNA devices. Re-starting those audio streams fixes the problem. See pa-dlna issue 15 and Wireplumber issue 511.
A workaround may be used with the `--clients-uuids` command line option, see the pa-dlna command documentation.

[2] The `pa-dlna` and `upnp-cmd` programs can be run simultaneously.

1.1.3 parec

pa-dlna uses the pulseaudio `parec` program[3]. Depending on the linux distribution it may be already installed as a dependency of pulseaudio or of pipewire-pulse. If not, then the package that owns `parec` must be installed. On archlinux the package name is `libpulse`, on debian it is pulseaudio-utils.

1.1.4 systemd

The python-systemd package is required to run the pa-dlna systemd service unit. It is packaged by almost all Linux distributions but under different names. To install the package from a Linux distribution or from PYPi, see the `Installation section` on the main page of the python-systemd git repository.

1.1.5 Encoders

No other dependency is required by pa-dlna when the DLNA devices support raw PCM L16 (**RFC 2586**)[4].

Optionally, encoders compatible with the audio mime types supported by the devices may be used. `pa-dlna` currently supports the ffmpeg (mp3, wav, aiff, flac, opus, vorbis, aac), the flac and the lame (mp3) encoders. The list of supported encoders, whether they are available on this host and their options, is printed by the command that prints the default configuration:

```
$ pa-dlna --dump-default
```

1.1.6 pavucontrol

Optionally, one may install the `pavucontrol` package for easier management of associations between sound sources and DLNA devices.

1.2 Installation

1.2.1 pipewire as a pulseaudio sound server

The `pipewire`, `pipewire-pulse` and `wireplumber` packages must be installed and the corresponding programs started. If you are switching from pulseaudio, make sure to remove `/etc/pulse/client.conf` or to comment out the setting of `default-server` in this file as pulseaudio and pipewire do not use the same unix socket path name.

The `parec` 's package includes the `pactl` program. One may check that the installation of pipewire as a pulseaudio sound server is successfull by running the command:

```
$ pactl info
```

1.2.2 pa-dlna

Install `pa-dlna` with pip:

```
$ python -m pip install pa-dlna
```

[3] The `parec` program also uses the `libpulse` library which is included in `parec` 's package or is installed as a dependency. Note also that this package includes the `pactl` and `pacmd` programs.

[4] DLNA devices must support the HTTP GET transfer protocol and must support HTTP 1.1 as specified by Annex A.1 of the ConnectionManager:3 Service UPnP specification.

1.3 Configuration

A `pa-dlna.conf` user configuration file overriding the default configuration may be used to:

- Change the preferred encoders ordered list used to select an encoder.
- Configure encoder options.
- Set an encoder for a given device and configure the options for this device.
- Configure the *sample_format*, *rate* and *channels* parameters of the `parec` program used to forward PulseAudio streams, for a specific device, for an encoder type or for all devices.

See the configuration section of the pa-dlna documentation.

CHAPTER

TWO

USAGE

2.1 *pa-dlna command* usage

In this section:

- A short description of *DLNA Networking* relevant to `pa-dlna`, the list of the UDP/TCP ports being used and what may be done when a firewall is in use.
- Events triggering *DLNA device discovery* and what happens then.
- Configuration of a *Source-sink association* between an application as a Pulseaudio source (music player, firefox, etc…) and a DLNA device.

The *pa-dlna command* section lists the pa-dlna command line options.

2.1.1 DLNA Networking

UPnP device discovery (and therefore DLNA device discovery) is implemented by two protocols that run independently:

1. To search for devices, an UPnP control point such as pa-dlna:
   - Send MSEARCH UDP multicast datagrams to `239.255.255.250:1900`.
   - Listen to the source IP address and **source UDP port** that is used to send the MSEARCH request for the responses that are sent by the devices.
2. To be notified of UPnP device advertisements, an UPnP control point listens on UDP port `1900` to receive NOTIFY UDP multicast datagrams broadcasted by the devices.

When pa-dlna is ready to forward a Pulseaudio stream to a DLNA device, it starts an HTTP server, if not already running, that listens on TCP port 8080 (the default) at the local IP address of the network that has been used to discover the DLNA device. This HTTP server only accepts connection requests from the IP addresses of DLNA devices that have been learnt by pa-dlna. The HTTP session is used to forward the Pulseaudio stream.

Ports that must be enabled on a network interface by a firewall:

- **MSEARCH UDP port:**
  This is the UDP port specified by the `--msearch-port` command line option of `pa-dlna`. This option may be used to set the specific **source UDP port**[1] of MSEARCH UDP datagrams so that this port may be enabled by a firewall. Otherwise if this option is not used or set to 0 the source port is chosen randomly by the operating system and it is necessary to configure the firewall to enable all UDP ports on the network interface.
- **NOTIFY UDP port:**
  The port value is set by the UPnP specifications as `1900`. When blocked by a firewall, UPnP device advertisements are not received but UPnP devices are still discovered with MSEARCH.

[1] Prefer choosing a port in the range 49152–65535.

- **HTTP server's TCP port:**
  This is the TCP port specified by the `--port` command line option of `pa-dlna`. The default is port `8080`.

2.1.2 DLNA device discovery

UPnP discovery is triggered by NICs[2] state changes. That is, whenever a configured NIC or the NIC of a configured IP address becomes up. Here are some examples of events triggering UPnP discovery on an IP address after `pa-dlna` or `upnp-cmd`[3] has been started:

- A wifi controller connects to a hotspot and acquires a new IP address through DHCP, possibly a different address from the previous one.
- A static IP address has been configured on an ethernet card connected to an ethernet switch and the switch is turned on.

`pa-dlna` registers a new sink with Pulseaudio upon the discovery of a DLNA device and selects an encoder (see the *Configuration* section for how the encoder is selected).

The sink appears in the `Output Devices` tab of the `pavucontrol` graphical tool and is listed by the `pactl` Pulseaudio commands.

2.1.3 Source-sink association

Pulseaudio remembers the association between a source and a sink across different sessions. A thorough description of this feature is given in "PulseAudio under the hood" at Automatic setup and routing.

Use `pavucontrol` or `pactl` to establish this association between a source and a DLNA device while the source is playing and the DLNA device has been registered with Pulseaudio. Establishing this association is needed only once.

With `pavucontrol`:
In the `Playback` tab, use the drop-down list of the source to select the DLNA sink registered by `pa-dlna`.

With `pactl`:
Get the list of sinks and find the index of the registered DLNA sink:

```
$ pactl list sinks | grep -e 'Sink' -e 'Name'
```

Get the list of sources and find the index of the source[4]; the source must be playing:

```
$ pactl list sink-inputs | grep -e 'Sink Input' -e 'binary'
```

Using both indexes create the association between the sink input and the DLNA sink registered by `pa-dlna`:

```
$ pactl move-sink-input <sink-input index> <sink index>
```

When the DLNA device is not registered (`pa-dlna` is not running or the DLNA device is turned off) Pulseaudio temporarily uses the default sink as the sink for this association. It is usually the host's sound card. See Default/fallback devices.

[2] Network Interface Controller.
[3] The list of the IP addresses learnt by pa-dlna through UPnP discovery may be listed with `upnp-cmd` by printing the value of the `ip_monitored` variable in the main menu.
[4] A source is called a sink-input by Pulseaudio.

2.2 *upnp-cmd command* usage

An interactive command line tool for introspection and control of UPnP devices.

The *upnp-cmd command* section lists the upnp-cmd command line options.

Some examples:

- When the UPnP device[5] is a DLNA device[6], running the `GetProtocolInfo` command in the `ConnectionManager` service menu prints the list of mime types supported by the device.
- Commands in the `RenderingControl` service allow to control the volume or mute the device.

Note: Upon `upnp-cmd` startup one must allow for the device discovery process to complete before being able to select a device.

Commands usage:

- Command completion and command arguments completion is enabled with the <Tab> key.
- Help on the current menu is printed by typing `?` or `help`.
- Help on one of the commands is printed by typing `help <command name>` or `? <command name>`.
- Use the arrow keys for command line history.
- When the UPnP device is a DLNA device and one is prompted for `InstanceID` by some commands, use one of the `ConnectionIDs` printed by `GetCurrentConnectionIDs` in the `ConnectionManager` service. This is usually `0` as most DLNA devices do not support `PrepareForConnection` and therefore support only one connection.
- To return to the previous menu, type `previous`.
- To exit the command type `quit`, `EOF`, <Ctl-d> or <Ctl-c>.

The menu hierarchy is as follows:

1. **Main menu prompt:**
   [Control Point]
2. **Next submenu prompt:**
   `friendlyName` of the selected device, for example [Yamaha RN402D].
3. **Next submenu prompt:**
   Either the service name when a service has been selected as for example [ConnectionManager] or `friendlyName` of the selected device when an embedded device has been selected.

One can select a DLNA device in the main menu and select a service or an embedded device in the device menu.

2.3 UPnP Library

UPnP devices are discovered by broadcasting MSEARCH SSDPs every 60 seconds (the default) and by handling the NOTIFY SSDPs broadcasted by the devices.

The `max-age` directive in MSEARCH responses and NOTIFY broadcasts refreshes the aging time of the device. The device is discarded of the list of registered devices when this aging time expires.

UPnP eventing is not supported.

[5] An UPnP device implements the UPnP Device Architecture specification.
[6] A DLNA device is an UPnP device and implements the MediaRenderer Device specification and the ConnectionManager, AVTransport and RenderingControl services.

CHAPTER

THREE

CONFIGURATION

The configuration is defined by the *Built-in Default Configuration*[1] overriden by the *User configuration* file if it exists. It is used in the following two stages of the `pa-dlna` process:

- The selection of the encoder and audio mime-type.
- The forwarding of an audio stream.

3.1 Encoder selection

`pa-dlna` fetches the DLNA device supported mime-types using the `GetProtocolInfo` UPnP command[2] and selects the first encoder/mime-type in the configured `selection` option that matches an item of the list returned by `GetProtocolInfo`.

If not already existing, an HTTP server is then started that answers requests on the local IP address where the DLNA device has been discovered.

3.2 Streaming

When PulseAudio (actually libpulse) notifies `pa-dlna` of the existence of a new stream from a source to the DLNA sink, it sends a `SetAVTransportURI` or `SetNextAVTransportURI`[3] UPnP SOAP[4] action to the device. This command holds:

- The stream metadata.
- The selected mime-type.
- The URL to be used by the device to fetch the stream by initiating an HTTP GET for this URL.

Upon responding to the HTTP GET request `pa-dlna` forks the `parec` process and the selected encoder process[5] using the configured options. The output of the `parec` process is piped to the encoder process, the output of the encoder process is written to the HTTP socket.

It is possible to test an encoder configuration without using a DLNA device with the help of the ffplay program from the ffmpeg suite and a tool that retrieves HTTP files such as curl or wget. Here is an example with the `L16Encoder`:

- Set the L16Encoder at the highest priority in the pa-dlna.conf file.
- Run pa-dlna with the `test-devices` command line option[6]:

[1] The default configuration is printed by the command: `` `$ pa-dlna --dump-default` ``
[2] The `GetProtocolInfo` command in the `ConnectionManager` service menu of the `upnp-cmd` command line tool prints this same list.
[3] The `SetNextAVTransportURI` is used when the `track_metadata` option is set.
[4] Simple Object Access Protocol. A remote-procedure call mechanism based on XML that sends commands and receives values over HTTP.
[5] Except when the audio/L16 mime type is selected.
[6] Note that the `;` character must be escaped on the command line or the value of the `--test-devices` option must be quoted.

```
$ pa-dlna --test-devices audio/L16\;rate=44100\;channels=2
```

- Start a music player application and play some track.
- Associate this source with the `DLNATest_L16 - 0ab65` DLNA sink in pavucontrol.
- Fetch the stream with curl as a file named `output` using the URL printed by the logs[7]:

```
$ curl http://127.0.0.1:8080/audio-content/uuid:e7fa8886-6d97-a009-b6b6-
↪6b1171b0ab65 -o output
```

- Play the `output` file with the command:

```
$ ffplay -f s16be -ac 2 -ar 44100 output
```

3.3 Encoders configuration

The encoders configuration is defined by the *Built-in Default Configuration* that may be overriden by the user's `pa-dlna.conf` file.

The `pa-dlna.conf` file also allows the specification of the encoder and its options for a given DLNA device with a section named [EncoderName.UDN]. In this case the selection of the encoder using the `selection` option is by-passed and EncoderName is the selected encoder. UDN is the udn[8] of the device as printed by the logs or by the `upnp-cmd` command line tool.

The default configuration is structured as an INI file, more precisely as text that may be parsed by the configparser Python module. The user's configuration file is also an INI file and obeys the same rules as the default configuration:

- A section is either [DEFAULT], [EncoderName] or [EncoderName.UDN]. The options defined in the [DEFAULT] section apply to all the other sections and are overriden when also defined in the [EncoderName] or [EncoderName.UDN] sections. There is an exception with the `selection` option that is only meaningful in a [DEFAULT] section and ignored in all the other sections.
- The `selection` option is an ordered comma separated list of encoders. This list is used to select the first encoder matching one of the mime-types supported by a discovered DLNA device when there is no specific [EncoderName.UDN] configuration for the given device.
- The options defined in the user's `pa-dlna.conf` file override the options of the default configuration.
- Section names and options are case sensitive.
- Boolean values are resticted to `yes` or `no`.

3.4 User configuration

The full path name of the user's `pa-dlna.conf` file is determined by `pa-dlna` as follows:

- If the `XDG_CONFIG_HOME` environment variable is set, the path name is `$XDG_CONFIG_HOME/pa-dlna/pa-dlna.conf`.
- Otherwise the path name is `$HOME/.config/pa-dlna/pa-dlna.conf`.

When `pa-dlna.conf` is not found, the program uses the default configuration. Otherwise it uses the default configuration with its options overriden by the user's configuration and with the added [EncoderName.UDN] sections.

Here is an example of a `pa-dlna.conf` file:

[7] DLNATest device sink names and URLs are built using the sha1 of the audio mime type and therefore are consistent across `pa-dlna` sessions.
[8] UDN: Unique Device Name. Universally-unique identifier of an UPnP device.

```
[DEFAULT]
selection =
    Mp3Encoder,
    FFMpegMp3Encoder,
    FFMpegFlacEncoder,

[FFMpegFlacEncoder]
track_metadata = no

[FFMpegMp3Encoder]
bitrate = 320

[FFMpegMp3Encoder.uuid:9ab0c000-f668-11de-9976-00a0de98381a]
```

In this example:

- The DLNA device whose udn is `uuid:9ab0c000-f668-11de-9976-00a0de98381a` uses the FFMpegMp3Encoder with the default bitrate.
- The other devices may use the three encoders of the selection, the preferred one being the Mp3Encoder with the default bitrate.
- The FFMpegMp3Encoder is only used if the Mp3Encoder (the lame encoder) is not available and in that case it runs with a bitrate of 320 Kbps.
- The FFMpegFlacEncoder is used when a DLNA device does not support the 'audio/mp3' and 'audio/mpeg' mime types and in that case its track_metadata option is not set.
- If a DLNA device does not support the mp3 or the flac mime types, then it cannot be used even though the device would support one of the other mime types defined in the overriden default configuration.

One can verify what is the actual configuration used by `pa-dlna` by running the program with the `--dump-internal` command line option. A Python dictionary is printed with keys being `EncoderName` or `UDN` and the values a dictionary of their options. The `EncoderName` keys are ordered according to the `selection` option.

3.5 PulseAudio options

Options used by the `parec` and encoder programs (see how those programs are used in the *Streaming* section):

sample_format
: The default value is `s16le`.

  The encoders supporting the `audio/L16` mime types (i.e. uncompressed audio data as defined by RFC 2586) have this option set to `s16be` as specified by the RFC and it cannot be modified by the user.

  See the Pulseaudio supported sample formats.

rate
: The Pulseaudio sample rate (default: 44100).

channels
: The number of audio channels (default: 2).

3.6 Common options

args
: The `args` option is the encoder program's command line. When the `args` option is None, the encoder command line is built from the Pulseaudio options and the encoder's specific options.

  As all the other options (except `sample_format` in some cases, see above) it may be overriden by the user.

track_metadata

- When `yes`, each track is streamed in its own HTTP session allowing the DLNA device to get each track meta data as described in the *Meta Data* section.

  This is the default.

- When `no`, there is only one HTTP session for all the tracks. Set this option to `no` when the logs show ERROR entries upon tracks changes.

soap_minimum_interval
: UPnP SOAP actions that start/stop a stream are spread out at `soap_minimum_interval` seconds to avoid the problem described at issue #16. This applies only to the SOAP actions that initiate or stop a stream: SetAVTransportURI, SetNextAVTransportURI and Stop.

  The default is 5 seconds.

3.7 Encoder specific options

Encoder specific options (for example `bitrate`) are listed in *Built-in Default Configuration* with their default value. They are used to build the encoder command line when `args` is None.

CHAPTER

FOUR

BUILT-IN DEFAULT CONFIGURATION

As printed by the command `pa-dlna --dump-default`.

```
# The pa-dlna default configuration.
#
# This is the built-in pa-dlna configuration written as text. It can be
# parsed by a Python Configuration parser and consists of sections, each led
# by a [section] header, followed by option/value entries separated by
# '='. See https://docs.python.org/3/library/configparser.html.
#
# The 'selection' option is written as a multi-line in which case all the
# lines after the first line start with a white space.
#
# The default value of 'selection' lists the encoders in this order:
#     - mp3 encoders first as mp3 is the most common encoding
#     - lossless encoders
#     - then lossy encoders
# See https://trac.ffmpeg.org/wiki/Encode/HighQualityAudio.

[DEFAULT]
selection =
    Mp3Encoder,
    FFMpegMp3Encoder,
    L16Encoder,
    FFMpegL16WavEncoder,
    FFMpegAiffEncoder,
    FlacEncoder,
    FFMpegFlacEncoder,
    FFMpegOpusEncoder,
    FFMpegVorbisEncoder,
    FFMpegAacEncoder,
sample_format = s16le
rate = 44100
channels = 2
track_metadata = yes
soap_minimum_interval = 5
args = None

[FFMpegAacEncoder]
# Aac encoder.
#
```

(continues on next page)

(continued from previous page)

```
# 'bitrate' is expressed in kilobits.
# See also https://trac.ffmpeg.org/wiki/Encode/AAC.
#
# available: yes
# pgm: /usr/bin/ffmpeg
# mime_types: ['audio/aac', 'audio/x-aac', 'audio/vnd.dlna.adts']
#
bitrate = 192
args = -loglevel error -hide_banner -nostats -ac 2 -ar 44100 -f s16le -i - -f adts
-c:a aac -b:a 192k pipe:1

[FFMpegAiffEncoder]
# Lossless Aiff Encoder.
#
# available: yes
# pgm: /usr/bin/ffmpeg
# mime_types: ['audio/aiff']
args = -loglevel error -hide_banner -nostats -ac 2 -ar 44100 -f s16le -i - -f aiff
pipe:1

[FFMpegFlacEncoder]
# Lossless Flac encoder.
#
# See also https://ffmpeg.org/ffmpeg-all.html#flac-2.
#
# available: yes
# pgm: /usr/bin/ffmpeg
# mime_types: ['audio/flac', 'audio/x-flac']
args = -loglevel error -hide_banner -nostats -ac 2 -ar 44100 -f s16le -i - -f flac
pipe:1

[FFMpegL16WavEncoder]
# Lossless PCM L16 encoder with a wav container.
#
# available: yes
# pgm: /usr/bin/ffmpeg
# mime_types: ['audio/l16']
#
sample_format = s16be
args = -loglevel error -hide_banner -nostats -ac 2 -ar 44100 -f s16be -i - -f wav
pipe:1

[FFMpegMp3Encoder]
# Mp3 encoder.
#
# Setting 'bitrate' to 0 causes VBR encoding to be chosen and 'qscale'
# to be used instead, otherwise 'bitrate' is expressed in kilobits.
# See also https://trac.ffmpeg.org/wiki/Encode/MP3.
#
# available: yes
# pgm: /usr/bin/ffmpeg
# mime_types: ['audio/mp3', 'audio/mpeg']
```

(continues on next page)

(continued from previous page)

```
#
bitrate = 256
qscale = 2
args = -loglevel error -hide_banner -nostats -ac 2 -ar 44100 -f s16le -i - -f mp3
-c:a libmp3lame -b:a 256k pipe:1

[FFMpegOpusEncoder]
# Opus encoder.
#
# See also https://wiki.xiph.org/Opus_Recommended_Settings.
#
# available: yes
# pgm: /usr/bin/ffmpeg
# mime_types: ['audio/opus', 'audio/x-opus']
#
bitrate = 128
args = -loglevel error -hide_banner -nostats -ac 2 -ar 44100 -f s16le -i - -f opus
-c:a libopus -b:a 128k pipe:1

[FFMpegVorbisEncoder]
# Vorbis encoder.
#
# Setting 'bitrate' to 0 causes VBR encoding to be chosen and 'qscale'
# to be used instead, otherwise 'bitrate' is expressed in kilobits.
# See also https://ffmpeg.org/ffmpeg-all.html#libvorbis.
#
# available: yes
# pgm: /usr/bin/ffmpeg
# mime_types: ['audio/vorbis', 'audio/x-vorbis']
#
bitrate = 256
qscale = 3.0
args = -loglevel error -hide_banner -nostats -ac 2 -ar 44100 -f s16le -i - -f ogg
-c:a libvorbis -b:a 256k pipe:1

[FlacEncoder]
# Lossless Flac encoder.
#
# See the flac home page at https://xiph.org/flac/
# See also https://xiph.org/flac/documentation_tools_flac.html
#
# pgm: /usr/bin/flac
# available: yes
# mime_types: ['audio/flac', 'audio/x-flac']
args = - --silent --channels 2 --sample-rate 44100 --sign signed --bps 16 --endian
little

[L16Encoder]
# Lossless PCM L16 encoder without a container.
#
# This encoder does not use an external program for streaming. It only uses
# the Pulseaudio parec program.
```

(continues on next page)

(continued from previous page)

```
# See also https://datatracker.ietf.org/doc/html/rfc2586.
#
# mime_types: ['audio/l16']
#
sample_format = s16be

[Mp3Encoder]
# Mp3 encoder from the Lame Project.
#
# See the Lame Project home page at https://lame.sourceforge.io/
# See lame command line options at
#     https://svn.code.sf.net/p/lame/svn/trunk/lame/USAGE
#
# pgm: /usr/bin/lame
# available: yes
# mime_types: ['audio/mp3', 'audio/mpeg']
#
bitrate = 256
quality = 0
args = -r -s 44.1 --signed --bitwidth 16 --little-endian -q 0 -b 256 -
```

CHAPTER

FIVE

UPNP-CMD COMMAND

5.1 Synopsis

upnp-cmd [*options*]

UPnP discovery is run on all the networks (except the loopbak interface `lo`) when the `--ip-addresses` and `--nics` command line arguments are not used or empty. Otherwise both arguments may be used indifferently or even jointly.

5.2 Options

-h, **--help**

Show this help message and exit.

--version, **-v**

Show program's version number and exit.

--ip-addresses IP_ADDRESSES, **-a** IP_ADDRESSES

IP_ADDRESSES is a comma separated list of the IPv4 addresses of the networks where UPnP devices may be discovered (default: `''`).

--nics NICS, **-n** NICS

NICS is a comma separated list of the names of network interface controllers where UPnP devices may be discovered, such as `wlan0,enp5s0` for example (default: `''`).

--msearch-interval MSEARCH_INTERVAL, **-m** MSEARCH_INTERVAL

Set the time interval in seconds between the sending of the MSEARCH datagrams used for device discovery (default: 60)

--ttl TTL

Set the IP packets time to live to TTL (default: 2).

--logfile PATH, **-f** PATH

Add a file logging handler set at `debug` log level whose path name is PATH.

--nolog-upnp, **-u**

Ignore UPnP log entries at `debug` log level.

--log-aio, **-y**

Do not ignore asyncio log entries at `debug` log level; the default is to ignore those verbose logs.

CHAPTER

SIX

PA-DLNA COMMAND

6.1 Synopsis

pa-dlna [*options*]

UPnP discovery is run on all the networks (except the loopbak interface `lo`) when the `--ip-addresses` and `--nics` command line arguments are not used or empty. Otherwise both arguments may be used indifferently or even jointly.

6.2 Options

-h, **--help**

Show this help message and exit.

--version, **-v**

Show program's version number and exit.

--ip-addresses IP_ADDRESSES, **-a** IP_ADDRESSES

IP_ADDRESSES is a comma separated list of the local IPv4 addresses of the networks where UPnP devices may be discovered (default: `''`).

--nics NICS, **-n** NICS

NICS is a comma separated list of the names of network interface controllers where UPnP devices may be discovered, such as `wlan0,enp5s0` for example (default: `''`).

--msearch-interval MSEARCH_INTERVAL, **-m** MSEARCH_INTERVAL

Set the time interval in seconds between the sending of the MSEARCH datagrams used for UPnP device discovery (default: 60).

--msearch-port MSEARCH_PORT, **-p** MSEARCH_PORT

Set the local UDP port for receiving MSEARCH response messages from UPnP devices, a value of `0` means letting the operating system choose an ephemeral port (default: 0).

--ttl TTL

Set the IP packets time to live to TTL (default: 2).

--port PORT

Set the TCP port on which the HTTP server handles DLNA requests (default: 8080).

--dump-default, **-d**

Write to stdout (and exit) the default built-in configuration.

--dump-internal, -i

Write to stdout (and exit) the configuration used internally by the program on startup after the pa-dlna.conf user configuration file has been parsed.

--clients-uuids PATH

PATH is the name of the file where are stored the associations between client applications and their DLNA device uuid. This is used to work around Wireplumber issue 511 on Pipewire.

Client applications names that play an audio stream are written by pa-dlna to PATH with the uuid of the DLNA device. In a next pa-dlna session and upon discovering a DLNA device, the list of the playback streams currently being currently run by the sound server is inspected by pa-dlna and if one of the client applications names matches an entry in PATH that maps to this DLNA device, then the playback stream is moved to the DLNA device by pa-dlna.

These associations can be removed from PATH or commented out by the user upon becoming irrelevant.

--loglevel {debug,info,warning,error}, **-l** {debug,info,warning,error}

Set the log level of the stderr logging console (default: info).

--systemd

Run as a systemd service unit.

--logfile PATH, **-f** PATH

Add a file logging handler set at debug log level whose path name is PATH.

--nolog-upnp, -u

Ignore UPnP log entries at debug log level.

--log-aio, -y

Do not ignore asyncio log entries at debug log level; the default is to ignore those verbose logs.

--test-devices MIME-TYPES, **-t** MIME-TYPES

MIME-TYPES is a comma separated list of distinct audio mime types. A DLNATestDevice is instantiated for each one of these mime types and registered as a virtual DLNA device. Mostly for testing.

CHAPTER

SEVEN

SYSTEMD

7.1 Usage

The python-systemd package is required to run the pa-dlna systemd service unit.

pa-dlna runs as a systemd/User service unit (Pulseaudio and Pipewire run also as a user service unit). Only one Control Point (such as pa-dlna) may interact with a given DLNA device and pa-dlna enforces this rule by allowing only one pa-dlna process per Sound Server.

Table 1: Systemd commands for pa-dlna

Purpose	Command
Enable pa-dlna and start it	`systemctl --user enable --now pa-dlna`
Disable pa-dlna and stop it	`systemctl --user disable --now pa-dlna`
Start pa-dlna	`systemctl --user start pa-dlna`
Stop pa-dlna	`systemctl --user stop pa-dlna`
Get the state of pa-dlna	`systemctl --user status pa-dlna`
Print the journal of pa-dlna	`journalctl --user -u pa-dlna`

7.2 The pa-dlna.service unit

The `pa-dlna.service` unit file is located in the `systemd` directory at the root of the pa-dlna git repository.

Its content is:

```
# When enabled, the pa-dlna service unit is started automatically after the
# pulseaudio or pipewire service unit is started. It will also stop when the
# pulseaudio or pipewire service unit stops. However it will stop when the
# pulseaudio or pipewire service unit is restarted but it will not start.
#
# Both pa-dlna and pulseaudio service units are of 'Type=notify'. This means
# that pa-dlna will only start after pulseaudio has notified systemd that it
# is ready and pa-dlna may connect successfully to libpulse.
#
# However the pipewire service unit is of 'Type=simple'. In that case and if
# pa-dlna fails to start with the error:
#       LibPulseStateError(('PA_CONTEXT_FAILED', 'Connection refused'))
# add a delay to the pa-dlna start up sequence with the directive:
#       ExecStartPre=/bin/sleep 1
#
# Any pa-dlna option may be added to the 'ExecStart' directive, for example to
```

(continues on next page)

(continued from previous page)

```
# restrict the allowed NICs or IP addresses (recommended) or to change the
# log level.
# The '--systemd' option is required.
#
# The 'python-systemd' package is required.

[Unit]
Description=Pa-dlna Service
Documentation=https://pa-dlna.readthedocs.io/en/stable/

After=pipewire-session-manager.service pulseaudio.service

[Service]
Type=notify
ExecStart=/usr/bin/pa-dlna --systemd
Slice=session.slice

NoNewPrivileges=yes
UMask=0077

[Install]
WantedBy=pipewire-session-manager.service pulseaudio.service
```

CHAPTER

EIGHT

DEVELOPMENT

8.1 Design

8.1.1 Meta Data

This feature is enabled on a per encoder or per device basis with the `track_metadata` option set to `yes`. It is enabled by default.

When `pa-dlna` receives a `change` event from pulseaudio and this event is related to a change to the meta data as for example when a new track starts with a new song, the following sequence of events occurs:

- `pa-dlna`:
  - Writes the last chunk to the HTTP socket (see Chunked Transfer Coding) and sends a `SetNextAVTransportURI` SOAP action with the new meta data.
  - Upon receiving the HTTP GET request from the device, instantiates a new Track and starts a task to run the pulseaudio stream.
- The DLNA device:
  - Gets the `SetNextAVTransportURI` with the new meta data and sends a GET request to start a new HTTP session for the next track while still playing the current track from its read buffer.
  - Still playing the current track, pre-loads the read buffer of the new HTTP session.
  - Upon receiving the last chunk for the current track, starts playing the next track.

This way, the last part of the current track is not truncated by the amount of latency introduced by the device's read buffer and the delay introduced by filling the read buffer of the next track is minimized.

8.1.2 Asyncio Tasks

Task names in **bold** characters indicate that there is one such task for each DLNA device, when in *italics* that there may be such tasks for each DLNA device.

UPnPControlPoint tasks:

ssdp notify	Monitor reception of NOTIFY SSDPs.
ssdp msearch	Send MSEARCH SSDPs at regular intervals.
root devices	Implement control of the aging of an UPnP root device.

AVControlPoint tasks:

main	Instantiate the UPnPControlPoint that starts the UPnP tasks. Create the pulse task, the http_server task, the renderer tasks. Create the shutdown task. Handle UPnP notifications.
pulse	Monitor pulseaudio sink-input events.
*maybe_*	Handle a `remove` pulse event.
http_se	Serve DLNA HTTP requests, one task per IP address. Start the client_connected tasks.
renderers	Act upon pulseaudio events. Run UPnP SOAP actions.
abort	Abort the pa-dlna program.
shutdown	Wait on event pushed by the signal handlers.

HTTPServer tasks:

client_conn	HTTPServer callback wrapped by asyncio in a task. Start the StreamSession tasks: `parec | encoder program | HTTP socket`.

StreamSession tasks:

parec process	Start the parec process and wait for its exit.
parec log_stderr	Log the parec process stderr.
encoder process	Start the encoder process and wait for its exit.
encoder log_stderr	Log the encoder process stderr.
track	Write the audio stream to the HTTP socket.

Track tasks:

shutdown	Write the last chunk and close the HTTP socket.

8.1.3 DLNA Device Registration

For a new DLNA device to be registered, `pa-dlna` must establish the **local** network address to be used in the URL that must be advertised to the DLNA device in the `SetAVTransportURI` and `SetNextAVTransportURI` SOAP actions, so that the DLNA device may initiate the HTTP session and start the streaming. This depends on which event triggered this registration:

Reception of the unicast response to an UPnP MSEARCH SSDP.
The destination address of the SSDP response is the address that is being looked for.

MSEARCH SSDP are sent by `pa-dlna` every 60 seconds (default).

Reception of an UPnP NOTIFY SSDP, broadcasted by the device[1].
The DLNA device can be registered only if the source address of this packet belongs to one of the subnets of the network interfaces. That is, the DLNA device and the host belong to the same subnet on this interface and the local IP address on this subnet is the address that is being looked for.

[1] All sockets bound to the notify multicast address receive the datagram sent by a DLNA device, even though it has been received by only one interface at the physical layer.

The UPnP Device Architecture specification does not specify the periodicity of NOTIFY SSDPs sent by DLNA devices.

8.2 Development process[2]

8.2.1 Requirements

Development:

- curl and ffmpeg are used by some tests of the test suite. When missing, those tests are skipped. curl is also needed when releasing a new version to fetch the GitLab test coverage badge.
- ffmpeg, the Upmpdcli DLNA Media Renderer, the MPD Music Player Daemon and a running Pulseaudio or PipeWire sound server are needed to run the tests of the `test_tracks` Python module (otherwise those tests are skipped).

  An audio track sourced by ffmpeg is streamed by pa-dlna to the Upmpdcli DLNA that outputs the stream to MPD, which in turn outputs the stream to a PulseAudio/PipeWire sink created by `test_tracks`. Monitoring the state of this sink allows checking that the audio track does follow this path. This scenario may be run at the debug log level with the following command:

```
$ python -m pa_dlna.tests.test_tracks [EncoderName]
```

- pactl is needed to run the tests that connect to the pulseaudio or pipewire sound server. When missing, those tests are skipped.
- docker may be used to run the test suite in a pulseaudio or pipewire debian container. Follow the instructions written as comments in each of the `Dockerfile.pulse` and `Dockerfile.pipewire` Docker files.
- coverage is used to get the test suite coverage.
- python-packaging is used to set the development version name as conform to PEP 440.
- flit is used to publish pa-dlna to PyPi and may be used to install pa-dlna locally.

  At the root of the pa-dlna git repository, use the following command to install pa-dlna locally:

```
$ flit install --symlink [--python path/to/python]
```

  This symlinks pa-dlna into site-packages rather than copying it, so that you can test changes by running the `pa-dlna` and `upnp-cmd` commands provided that the `PATH` environment variable holds `$HOME/.local/bin`.

  Otherwise without using flit, one can run those commands from the root of the repository as:

```
$ python -m pa_dlna.pa_dlna
$ python -m pa_dlna.upnp_cmd
```

Documentation:

- Sphinx[3].
- Read the Docs theme.
- Building the pdf documentation:
  - The latex texlive package group.
  - Imagemagick version 7 or more recent.

[2] The shell commands in this section are all run from the root of the repository.
[3] Required versions at `docs/requirements.txt`.

8.2.2 Documentation

To build locally the documentation follow these steps:

- Generate the `default-config.rst` file:

```
$ python -m tools.gendoc_default_config
```

- Fetch the GitLab test coverage badge:

```
$ curl -o images/coverage.svg "https://gitlab.com/xdegaye/pa-dlna/badges/master/
↪coverage.svg?min_medium=85&min_acceptable=90&min_good=90"
$ magick images/coverage.svg images/coverage.png
```

- Build the html documentation and the man pages:

```
$ make -C docs clean html man latexpdf
```

8.2.3 Updating development version

Run the following commands to update the version name at latest documentation after a bug fix or a change in the features:

```
$ python -m tools.set_devpt_version_name
$ make -C docs clean html man latexpdf
$ git commit -m "Update development version name"
$ git push
```

8.2.4 Releasing

- Run the test suite from the root of the project[4]:

```
$ python -m unittest --verbose --catch --failfast
```

- Get the test suite coverage:

```
$ coverage run --include="./*" -m unittest
$ coverage report -m
```

- Update `__version__` in pa_dlna/__init__.py.
- When this new release depends on a more recent libpulse release than previously:
- Update `MIN_LIBPULSE_VERSION` in pa_dlna/__init__.py.
- Update the minimum required libpulse version in pyproject.toml.
- Update docs/source/history.rst if needed.
- Build locally the documentation, see one of the previous sections.
- Commit the changes:

```
$ git commit -m 'Version 1.n'
$ git push
```

- Tag the release and push:

[4] See unittest command line options.

```
$ git tag -a 1.n -m 'Version 1.n'
$ git push --tags
```

- Publish the new version to PyPi:

```
$ flit publish
```

CHAPTER

NINE

RELEASE HISTORY

Version 1.2.1

- Remove `flit_core` version upper limit in `pyproject.toml`.

Version 1.2

- Support Python version 3.14:
  - Fix warnings upon return/break/continue that exit a finally block (PEP 765).
  - Restrict the configparser delimiters set to `'='` when writing the default configuration file to fix the check (made starting with Python 3.14) that triggers configparser.InvalidWriteError: *Cannot write key that contains the* `':'` *delimiter*. See Cpython PR #129270.
- Fix a crash upon a race condition that occurs when the DLNA device is about to be closed and is simultaneously discovered again by an SSDP notify/msearch datagram.
- The UPnP device is now closed instead of being permanently disabled after a connection error ocurring while pa-dlna is streaming a track. It will be discovered again by the next SSDP_NOTIFY or SSDP_MSEARCH udp datagram after the connection is up again (issue #55).

Version 1.1

- Fix the `test_libpulse` deadlock in `pytest` when another pa-dlna instance is already running. The corresponding test cases are skipped.
- Fix the `test_main` failure and the warning about `TestEncoder` when the test suite is run by `pytest`.
- A run of the test suite aborts and prints a clear error message when the `libpulse` version in use is invalid (issue #52).

Version 1.0

- Ignore spaces in the `max-age` setting of the `CACHE-CONTROL` field of SSDP_NOTIFY datagrams (issue #50).

Version 0.16

- The required libpulse version is now `0.7` after the error handling changes made in the libpulse release.
- Fix music player on KDE randomly raises exception while switching to next track (issue #49).
- KDE music players (Juk, Elisa, Strawberry) misbehave by sending `remove` pulse events just before switching to a next track. A work-around to this problem using a timer is implemented that discards those events (issue #48).
- **[Sonos]** Accept HTTP 1.1 chunked encoding response to pa-dlna HTTP 1.0 requests.

Version 0.15

- The `Transfer-Encoding` HTTP 1.1 header in response to HTTP 1.0 GET requests is not supported (issue #47).
- Ignore invalid subelements in `Icons` (issue #40).
- Use `friendlyName`, the name displayed by pavucontrol, as Renderer's name.
- Add the pa-dlna systemd service unit.
- Fix L16Encoder failing to set the correct mime type when the `ProtocolInfo <contentFormat>` entry is simply `audio/L16` without the rate parameter (issue #36).
- Added a test framework that runs tests with Upmpdcli (a software DLNA MediaRenderer) and MPD on the PulseAudio or Pipewire sound server.
- A pdf document is part of the pa-dlna documentation. To access the documentation as a pdf document one must click on the icon at the down-right corner of any page of the documentation on the web. It allows to switch between stable and latest versions and to select the corresponding pdf document.
- Fix the development version name as PEP 440 conformant (issue #33).

Version 0.14

- pa-dlna versioning conforms to PEP 440.
- Exit with an error message when the `libpulse` version is older than the required one. The required libpulse version is currently `0.5`.
- **[Upmpdcli]** Fix cannot play on `upmpdcli` tracks whose metadata includes the & character (issue #30).
- Add the `--clients-uuids` command line option that may be used as a work around to Wireplumber issue 511 (issue #15).

Version 0.13

- The backtraces of unhandled exceptions that occur in asyncio tasks are logged at the debug log level. Otherwise these exceptions are just logged as an error with a message saying that the backtrace can be obtained by running the program at the debug log level.
- **[Moode UPNP]** Fix libexpat called by `upmpdcli` fails parsing the DIDL-Lite xml strings (issue #29).

Version 0.12

- Rename LibPulse.get_events() to get_events_iterator(). The change has been introduced by version 0.4 of the libpulse package (issue #26).
- Handle exceptions raised while getting the sink after `module-null-sink` has been loaded.
- Fix a typo in the installation documentation.

Version 0.11

- Import the libpulse package from Pypi.
- Support Python version 3.12 - Fix some network mock tests by forcing the release of control to the asyncio loop.

Version 0.10

- **[Teufel 3sixty]** Handle HTTP HEAD requests from DLNA devices. Some renderers fetch stream meta data via HEAD request before requesting actual media streams.
- Fix crash upon parsing empty deviceList in device description.

Version 0.9

- Support Pipewire version 1.0 and the previous version 0.3.

- Log the name of the sound server and its version.

Version 0.8

- Changing the volume level with `pavucontrol` does not interfere with the current audio stream.
- **[Marantz NR1200]** Support multiple embedded MediaRenderers in a DLNA device.
- The `deviceList` attribute of UPnPDevice is now a list instead of a dictionary.
- Do not age an UPnP root device upon receiving a `CACHE-CONTROL` header with a value set to `max-age=0`.

Version 0.7

- Name `libpulse` the Python package, interface to the `libpulse` library.
- Document which TCP/UDP ports may not be blocked by a firewall.
- Add the `--msearch-port` command line option.
- Tests are run in GitLab CI/CD with Pulseaudio and with Pipewire.
- Add Docker files to allow running the test suite in Pulseaudio and Pipewire debian containers.
- Update the README with package requirements for linux distributions.
- The `psutil` Python package must be installed now separately as this package is included by many distributions (debian, archlinux, fedora, ...).
- Log the sound server name and version.

Version 0.6

- **[Yamaha RN402D]** Spread out UPnP SOAP actions that start/stop a stream (issue #16).
- Fix the `args` option in the [EncoderName.UDN] section of the user configuration is always None.
- Log a warning when the sink-input enters the `suspended` state.
- Fix assertion error upon `exit` Pulseaudio event (issue #14).
- Support PipeWire. No change is needed to support PipeWire. The test suite runs successfully on PipeWire.
- Fix no sound when pa-dlna is started while the track is already playing (issue #13).
- Use the built-in libpulse package that uses ctypes to interface with the libpulse library and remove the dependency to `pulsectl_asyncio`.
- Wait for the http server to be ready before starting the renderer task. This also fixes the test_None_nullsink and test_no_path_in_request tests on GitLab CI/CD (issue #12).
- Support Python 3.11.

Version 0.5

- Log a warning upon an empty body in the HTTP response from a DLNA device (issue #11).
- UPnP discovery is triggered by NICs[1] state changes (issue #10).
- Add the `--ip-addresses`, `-a` command line argument (issue #9).
- Fix changing the `args` encoder option is ignored (issue #8).

Version 0.4

- `sample_format` is a new encoder configuration option (issue #3).
- The encoders sample format is `s16le` except for the `audio/l16` encoder (issue #7).

[1] Network Interface Controller.

- The encoder command line is now updated with `pa-dlna.conf` user configuration (issue #6).
- Fix the parec command line length keeps increasing at each new track when the encoder is set to track metadata (issue #5).
- Fix failing to start a new stream session while the device is still playing when the encoder is set to not track metadata (issue #4).
- Fix `pa-dlna` hangs when one types <Control-S> in the terminal where the program has been started (issue #2).

Version 0.3

- The test coverage of `pa-dlna` is 95%.
- UPnPControlPoint supports now the context manager protocol, not the asynchronous one.
- UPnPControlPoint.get_notification() returns now QUEUE_CLOSED upon closing.
- Fix some fatal errors on startup that were silent. Here are the missing error messages that are now printed when one of those fatal errors occurs:
  - Error: No encoder is available.
  - Error: The pulseaudio 'parec' program cannot be found.
- Fix curl: (18) transfer closed with outstanding read data remaining.
- Fix a race condition upon the reception of an SSDP msearch response that occurs just after the reception of an SSDP notification and while the instantiation of the root device is not yet complete.
- Failure to set SSDP multicast membership is reported only once.

Version 0.2

- Test coverage of the UPnP package is 94%.
- Fix unknown UPnPXMLFatalError exception.
- The `description` commands of `upnp-cmd` don't prefix tags with a namespace.
- Fix the `description` commands of `upnp-cmd` when run with Python 3.8.
- Fix IndexError exception raised upon OSError in network.Notify.manage_membership().
- Fix removing multicast membership when the socket is closed.
- Don't print a stack traceback upon error parsing the configuration file.
- Abort on error setting the file logging handler with `--logfile PATH`.

Version 0.1

- Publish the project on PyPi.

INDEX

Symbols

--clients-uuids
    command line option, 20
--dump-default
    command line option, 19
--dump-internal
    command line option, 19
--help
    command line option, 17, 19
--ip-addresses
    command line option, 17, 19
--log-aio
    command line option, 17, 20
--logfile
    command line option, 17, 20
--loglevel
    command line option, 20
--msearch-interval
    command line option, 17, 19
--msearch-port
    command line option, 19
--nics
    command line option, 17, 19
--nolog-upnp
    command line option, 17, 20
--port
    command line option, 19
--systemd
    command line option, 20
--test-devices
    command line option, 20
--ttl
    command line option, 17, 19
--version
    command line option, 17, 19
-a
    command line option, 17, 19
-d
    command line option, 19
-f
    command line option, 17, 20
-h
    command line option, 17, 19
-i
    command line option, 19
-l
    command line option, 20
-m
    command line option, 17, 19
-n
    command line option, 17, 19
-p
    command line option, 19
-t
    command line option, 20
-u
    command line option, 17, 20
-v
    command line option, 17, 19
-y
    command line option, 17, 20

C

command line option
    --clients-uuids, 20
    --dump-default, 19
    --dump-internal, 19
    --help, 17, 19
    --ip-addresses, 17, 19
    --log-aio, 17, 20
    --logfile, 17, 20
    --loglevel, 20
    --msearch-interval, 17, 19
    --msearch-port, 19
    --nics, 17, 19
    --nolog-upnp, 17, 20
    --port, 19
    --systemd, 20
    --test-devices, 20
    --ttl, 17, 19
    --version, 17, 19
    -a, 17, 19
    -d, 19
    -f, 17, 20

-h, 17, 19
-i, 19
-l, 20
-m, 17, 19
-n, 17, 19
-p, 19
-t, 20
-u, 17, 20
-v, 17, 19
-y, 17, 20

R

RFC
RFC 2586, 2