gerbera Documentation

Release 1.5.0

Gerbera Contributors

May 02, 2020

Contents

Gerbera is an UPnP Media Server.

It allows you to stream your digital media through your home network and listen to/watch it on a variety of UPnP compatible devices.

Gerbera should work with any UPnP compliant client, please tell us if you experience difficulties with particular models, also take a look at the *Supported Devices* list for more information.

- Browse and playback your media via your network on all kinds of devices.
- Metadata extraction from MP3, OGG, AAC, M4A, FLAC, JPG (and many more!) files.
- Media thumbnail support
- Web UI with a tree view of the database and the file system, allowing to add/remove/edit/browse your media
- Highly flexible media format transcoding via plugins / scripts
- Automatic directory rescans (timed, inotify)
- User defined server layout based on extracted metadata
- Supports last fm scrobbing using lastfmlib
- On the fly video thumbnail generation with libffmpegthumbnailer
- Support for external URLs (create links to internet content and serve them via UPnP to your renderer)
- runs on Linux, FreeBSD, NetBSD, Mac OS X, eCS
- runs on x86, Alpha, ARM, MIPS, Sparc, PowerPC

CHAPTER 1

UPnP

Gerbera implements the UPnP MediaServer V 1.0 specification that can be found on http://www.upnp.org/. The current implementation focuses on parts that are required by the specification, however we look into extending the functionality to cover the optional parts of the spec as well.

Universal Plug and Play (UPnP) is a set of networking protocols that permits networked devices, such as PCs, printers, mobile phones TVs, stereos, amplifiers and more to seamlessly discover each other's presence on the network and network services for data sharing,

CHAPTER 2

Legal

THIS SOFTWARE COMES WITH ABSOLUTELY NO WARRANTY! USE AT YOUR OWN RISK!

CHAPTER 3

Copyright

CHAPTER 4

License

Gerbera is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation. Gerbera is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details. You should have received a copy of the GNU General Public License version 2 along with Gerbera; if not, write to the **Free Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301, USA.**

4.1 Acknowledgments

We are using the following code in our tree:

- md5 implementation by L. Peter Deutsch <ghost at aladdin dot com>, Copyright (c) 1999 Aladdin Enterprises. All rights reserved. (See source headers for further details)
- md5 javascript implementation distributed under BSD License, Copyright (c) Paul Johnston 1999 - 2002. http://pajhome.org.uk/crypt/md5
- Prototype JavaScript Framework http://www.prototypejs.org/ (c) 2005-2007 Sam Stephenson, MIT-style license.
- (heavily modified version of) NanoTree http://nanotree.sourceforge.net/ (c) 2003 (?) Martin Mouritzen <martin at nano dot dk>; LGPL
- IE PNG fix from http://webfx.eae.net/dhtml/pngbehavior/pngbehavior.html
- The the Inotify::nextEvent() function is based on code from the inotify tools package, http://inotify-tools.sf.net/, distributed under GPL v2, (c) Rohan McGovern <rohan at mcgovern dot id dot au>

4.2 Contributions

- Gerbera is built on MediaTomb 0.12.1 GIT
- Initial version of the MediaTomb start up script was contributed by Iain Lea <iain at bricbrac dot de>

- TagLib support patch was contributed by Benhur Stein <benhur.stein at gmail dot com>
- ffmpeg metadata handler was contributed by Ingo Preiml <ipreiml at edu dot uni-klu dot ac dot at>
- ID3 keyword extraction patch was contributed by Gabriel Burca <gburca-mediatomb at ebixio dot com
- Photo layout by year/month was contributed by Aleix Conchillo Flaqué <aleix at member dot fsf dot org>
- lastfmlib patch was contributed by Dirk Vanden Boer <dirk dot vdb at gmail dot com>
- NetBSD patches were contributed by Jared D. McNeill <jmcneill at NetBSD dot org>

4.2.1 Installing Gerbera

We strongly advise using the packages provided by your distribution if available. Please see below for guides on how to install Gerbera on various distributions.

Docker

Docker images are provided on Docker Hub, we recommend running a tagged version.

```
docker pull gerbera/gerbera
```

Ubuntu/Mint

Stephen Czetty maintains a Ubuntu PPA.

```
sudo add-apt-repository ppa:stephenczetty/gerbera-updates
sudo apt-get update
sudo apt install gerbera
```

Gentoo

The latest version and live ebuild are in the main portage tree.

```
emerge -va net-misc/gerbera
```

Arch

Gerbera is available in AUR with both stable or git versions.

Fedora

Gerbera is available in Fedora 29 or later.

```
sudo dnf install gerbera
```

CentOS

Gerbera 1.2 for Centos x86/64 is available via GitHub: https://github.com/lukesoft76/CENTOS_7.

All necessary rpm files are listed in the provided github project https://github.com/lukesoft76/CENTOS_7 .

Attention! So far, Gerbera is not part of any repository that is maintained in CentOS 7 due to the fact that Gerbera is only available for Fedora 28 which is not the base for CentOS 7!

Debian

Gerbera is included in Buster and Sid.

```
sudo apt install gerbera
```

Due to the stable nature of Debian, these packages are likely to be some versions behind the current Gerbera release.

Deb-Multimedia.org also provide builds for Buster and Sid.

openSUSE

Gerbera is available on software.opensuse.org.

Entware (Optware)

Gerbera is available in Entware for your embedded device/router!

macOS

Gerbera is available as the Gerbera Homebrew Tap on macOS.

4.2.2 Run Gerbera

Note: Whilst you can run Gerbera as a "regular" application. It is strongly recommended to run it as a *system service* instead.

Warning: The server has an integrated *file system browser* in the UI, that means that anyone who has access to the UI can browse your file system (with user permissions under which the server is running) and also download your files! If you want maximum security - disable the UI. *Account authentication* offers simple protection that might hold back your kids, but it is not secure enough for use in an untrusted environment!

Note: Since the server is meant to be used in a home LAN environment the UI is enabled by default and accounts are deactivated, thus allowing anyone on your network to connect to the user interface.

First Time Launch

If you decide against running as a system service for whatever reason, then when run by a user the first time startup of Gerbera creates a folder called `~/.config/gerbera` in your home directory.

You must generate a `config.xml` file for Gerbera to use.

Review the *Generating Configuration* section of the documentation to see how to use `gerbera` to create a default configuration file.

Multiple Instances

If you want to run a second server from the same PC, make sure to use a different configuration file with a different udn and a different database.

After server launch the bookmark file is created in the `~/.config/gerbera` directory. You now can manually add the bookmark `~/.config/gerbera/gerbera.html` in your browser. This will redirect you to the UI if the server is running.

Assuming that you enabled the UI, you should now be able to get around quite easily.

Network Setup

Some systems require a special setup on the network interface. If Gerbera exits with UPnP Error -117, or if it does not respond to M-SEARCH requests from the renderer (i.e. Gerbera is running, but your renderer device does not show it) you should try the following settings (the lines below assume that Gerbera is running on a Linux machine, on network interface eth1):

```
# route add -net 239.0.0.0 netmask 255.0.0.0 eth1
# ifconfig eth1 allmulti
```

Those settings will be applied automatically by the init.d startup script.

You should also make sure that your firewall is not blocking port UDP port `1900` (required for SSDP) and UDP/TCP port of Gerbera. By default Gerbera will select a free port starting with `49152`, however you can specify a port of your choice in the configuration file.

Using Sqlite Database (Default)

By default Gerbera will use an SQLite database, it requires no configuration - you are ready to go! The database file will be created automatically and will be located `~/.config/gerbera/gerbera.db` If needed you can adjust the database file name and location in the server configuration file.

Using MySQL Database

If Gerbera was compiled with support for both databases, sqlite will be chosen as default because the initial database can be created and used without any user interaction. If Gerbera was compiled only with MySQL support, the appropriate config.xml file will be created in the `~/.config/gerbera` directory, but the server will then terminate, because user interaction is required.

Gerbera has to be able to connect to the MySQL server and at least the (empty) database has to exist. To create the database and provide Gerbera with the ability to connect to the MySQL server you need to have the appropriate permissions. Note that user names and passwords in MySQL have nothing to do with UNIX accounts, MySQL has it's own user names/passwords. Connect to the MySQL database as "root" or any other user with the appropriate permissions:

```
$ mysql [-u <username>] [-p]
```

(You'll probably need to use "-u" to specify a different MySQL user and "-p" to specify a password.)

Create a new database for Gerbera: (substitute "<database name>" with the name of the database)

```
mysql> CREATE DATABASE <database name>;
```

(You can also use "mysqladmin" instead.)

Give Gerbera the permissions to access the database:

```
mysql> GRANT ALL ON <database name>.*
       TO '<user name>'@'<hostname>'
       IDENTIFIED BY '<password>';
```

If you don't want to set a password, omit `IDENTIFIED BY` completely. You could also use the MySQL "root" user with Gerbera directly, but this is not recommended.

To create a database and a user named **gerbera** (who is only able to connect via `localhost`) without a password (the defaults) use:

```
mysql> CREATE DATABASE gerbera;
mysql> GRANT ALL ON gerbera.* TO 'gerbera'@'localhost';
```

If Gerbera was compiled with database auto creation the tables will be created automatically during the first startup. All table names have a `mt_` prefix, so you can theoretically share the database with a different application. However, this is not recommended.

If database auto creation was not compiled in you have to create the tables manually:

```
$ mysql [-u <username>] [-p] \
  <database name> < \
  <install prefix>/share/gerbera/mysql.sql
```

After creating the database and making the appropriate changes in your Gerbera config file you are ready to go - launch the server, and everything should work.

Command Line Options

Note: Command line options override settings in the configuration file

There is a number of options that can be passed via command line upon server start up, for a short summary you can invoke Gerbera with the following parameter:

```
$ gerbera --help
```

IP Address

```
--ip or -i
```

The server will bind to the given IP address, currently we can not bind to multiple interfaces so binding to `0.0.0.0` is not be possible.

Interface

```
--interface or -e
```

Interface to bind to, for example eth0, this can be specified instead of the IP address.

Port

```
--port or -p
```

Specify the server port that will be used for the web user interface, for serving media and for UPnP requests, minimum allowed value is `49152`. If this option is omitted a default port will be chosen, however, in this case it is possible that the port will change upon server restart.

Configuration File

```
--config or -c
```

By default Gerbera will search for a file named **config.xml** in the `~/.config/gerbera` directory. This option allows you to specify a config file by the name and location of your choice. The file name must be absolute.

Home Directory

```
--home or -m
```

Specify an alternative home directory. By default Gerbera will try to retrieve the users home directory from the environment, then it will look for a `.config/gerbera` directory in users home. If `.config/gerbera` was found the system tries to find the default configuration file (config.xml), if not found the system creates both, the `.config/gerbera` directory and the default config file.

This option is useful in two cases: when the home directory can not be retrieved from the environment (in this case you could also use -c to point Gerbera to your configuration file or when you want to create a new configuration in a non standard location (for example, when setting up daemon mode). In the latter case you can combine this parameter with the parameter described in ?

Config Directory

```
--cfgdir or -f
```

The default configuration directory is combined out of the users home and the default that equals to `.config/gerbera`, this option allows you to override the default directory naming. This is useful when you want to setup the server in a nonstandard location, but want that the default configuration to be written by the server.

Add Content

```
--add-file /path/to/file [--add-file /path/to/other/file]
```

Add the specified directory or file name to the database without UI interaction. The path must be absolute, if path is a directory then it will be added recursively. If path is a file, then only the given file will be imported. Can be supplied multiple times to add multiple paths

Log To File

```
--logfile or -l
```

Do not output log messages to stdout, but redirect everything to a specified file.

Debug Output

```
--debug or -D
```

Enable debug log output.

Compile Info

```
--compile-info
```

Print the configuration summary (used libraries and enabled features) and exit.

Version Information

```
--version
```

Print version information and exit.

Display Command Line Summary

```
--help or -h
```

Print a summary about the available command line options.

4.2.3 Gerbera Daemon

You can setup Gerbera to run as a **daemon** background system process

Using systemd

This outlines steps of how to add the Gerbera runtime as a system daemon using the **Systemd**. The gerbera installation uses cmake to configure and install the systemd daemon for gerbera. Default install path is `/etc/systemd/system/gerbera.service`

Create a Gerbera System User

You should run Gerbera as a separate user to avoid vulnerabilities in exposing root access.

Here is a way to create a system user in linux command line

```
$ sudo useradd --system gerbera
```

Verify that the `gerbera` user was created

```
$ id -u gerbera
```

Returns the user id of the user

Set Gerbera Permissions

The **gerbera** user must have access to `config.xml` file and to the directory referenced as the gerbera home. For example `/etc/gerbera`

```
$ sudo mkdir /etc/gerbera
$ sudo chown -Rv gerbera:gerbera /etc/gerbera
```

Enable Systemd Daemon

The cmake installation adds `/etc/systemd/system/gerbera.service` service file by default.

The installation does **not** enable the service daemon.

1. Notify `systemd` that a new gerbera.service file exists by executing the following command:

   ```
   $ sudo systemctl daemon-reload
   ```

2. Start up the daemon

   ```
   $ sudo systemctl start gerbera
   ```

Success

Check the status of gerbera. You should see success similar to below

```
$ sudo systemctl status gerbera

 gerbera.service - Gerbera Media Server
   Loaded: loaded (/etc/systemd/system/gerbera.service; disabled; vendor preset:␣
↪disabled)
   Active: active (running) since Wed 2017-09-20 19:48:44 EDT; 47s ago
 Main PID: 4818 (gerbera)
    Tasks: 12 (limit: 4915)
```

(continues on next page)

(continued from previous page)

```
   CGroup: /system.slice/gerbera.service
           └─4818 /usr/local/bin/gerbera -c /etc/gerbera/config.xml
```

Troubleshooting

If for some reason the service fails to start. You can troubleshoot the behavior by starting gerbera from the shell

```
$ su gerbera
Password:
bash-4.4$  /usr/local/bin/gerbera -c /etc/gerbera/config.xml

2017-09-20 19:54:47    INFO: Gerbera UPnP Server version 1.1.0_alpha - http://gerbera.
↪io/
2017-09-20 19:54:47    INFO:␣
↪================================================================================
2017-09-20 19:54:47    INFO: Gerbera is free software, covered by the GNU General␣
↪Public License version 2
2017-09-20 19:54:47    INFO: Copyright 2016-2017 Gerbera Contributors.
2017-09-20 19:54:47    INFO: Gerbera is based on MediaTomb: Copyright 2005-2010 Gena␣
↪Batsyan, Sergey Bostandzhyan, Leonhard Wimmer.
2017-09-20 19:54:47    INFO:␣
↪================================================================================
2017-09-20 19:54:47    INFO: Loading configuration from: /etc/gerbera/config.xml
2017-09-20 19:54:47    INFO: Checking configuration...
```

Using solaris

You can use the solaris script provided in `scripts/solaris` to add Gerbera as a service in solaris.

Using launchd

launchd is the daemon engine in macOS.

Create new Launch Agent

Use the `scripts/gerbera.io.plist` as a starting point. Save to user's launch agent path –>

`~/Library/LaunchAgents/gerbera.io.plist`

Load the Launch Agent

```
$ launchctl load ~/Library/LaunchAgents/gerbera.io.plist
```

Start the Launch Agent

```
$ launchctl start gerbera.io
```

Stop the Launch Agent

```
$ launchctl stop gerbera.io
```

4.2.4 Gerbera UI

The Gerbera application provides a web browser user interface. When you launch the Gerbera application the system reports URL to the user interface. The interface is available for you to maintain your media library in Gerbera.

Enable/Disable

The Gerbera UI is enabled or disabled using the *config.xml* file.

```
<config>
  <server>
    <ui enabled="no" show-tooltips="yes">
      <accounts enabled="no" session-timeout="30">
        <account user="" password=""/>
      </accounts>
    </ui>
  </server>
</config>
```

When the Gerbera server starts successfully it reports the location of the web browser user interface

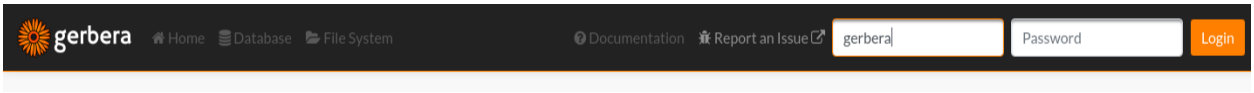
```
2018-01-28 17:05:28    INFO: The Web UI can be reached by following this link: http://
↪127.0.0.1:49152/
```

Login/Logout

When the *accounts* section of the server configuration is enabled you can login to the UI with the associated user and password.

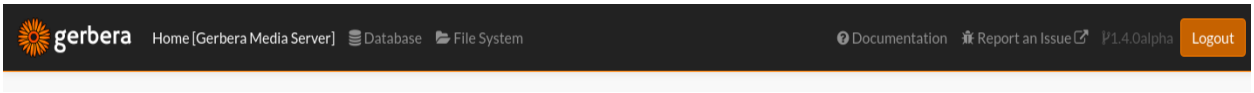
The UI supports entering the **username** and **password** in the top menu.

Note The system performs simple encoding of the password sent over HTTP to the Gerbera server. Do not consider the UI a completely secure data transmission.



If you choose to have the *accounts* disabled, then the UI automatically logs in and loads a new user session.

The menu is activated with successful login and you can choose from the following menu options:



- **Home** *Clears the view*
- **Database** *Loads the Gerbera database*
- **Filesystem** *Loads the local filesystem tree*

- **Report an Issue** *Opens URL to Gerbera's GitHub Issues*
- **Leave Beta** *Return to old frameset UI*

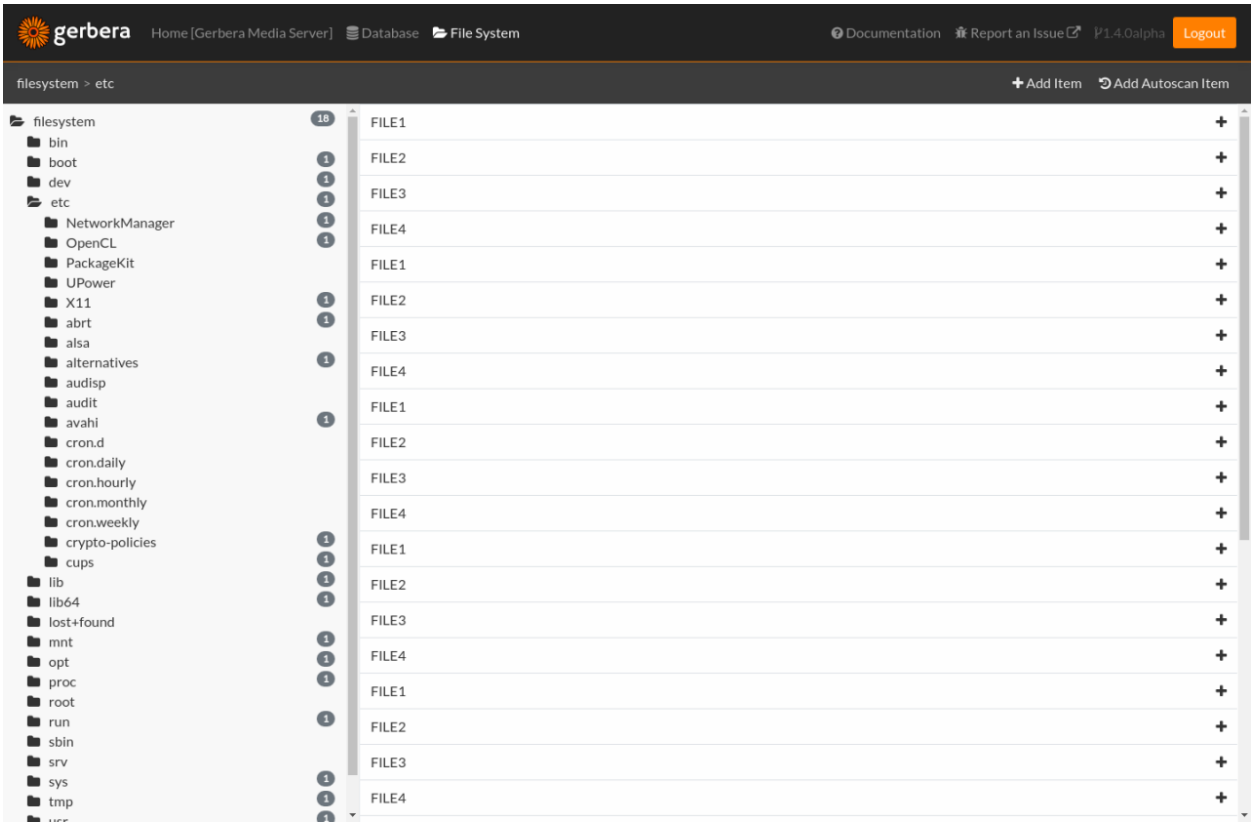
Features

The Gerbera web UI has several features to maintain your media.

- Filesystem Items View
- Database Items View
- Item Operations
- Trail Operations
- Notifications

Filesystem View

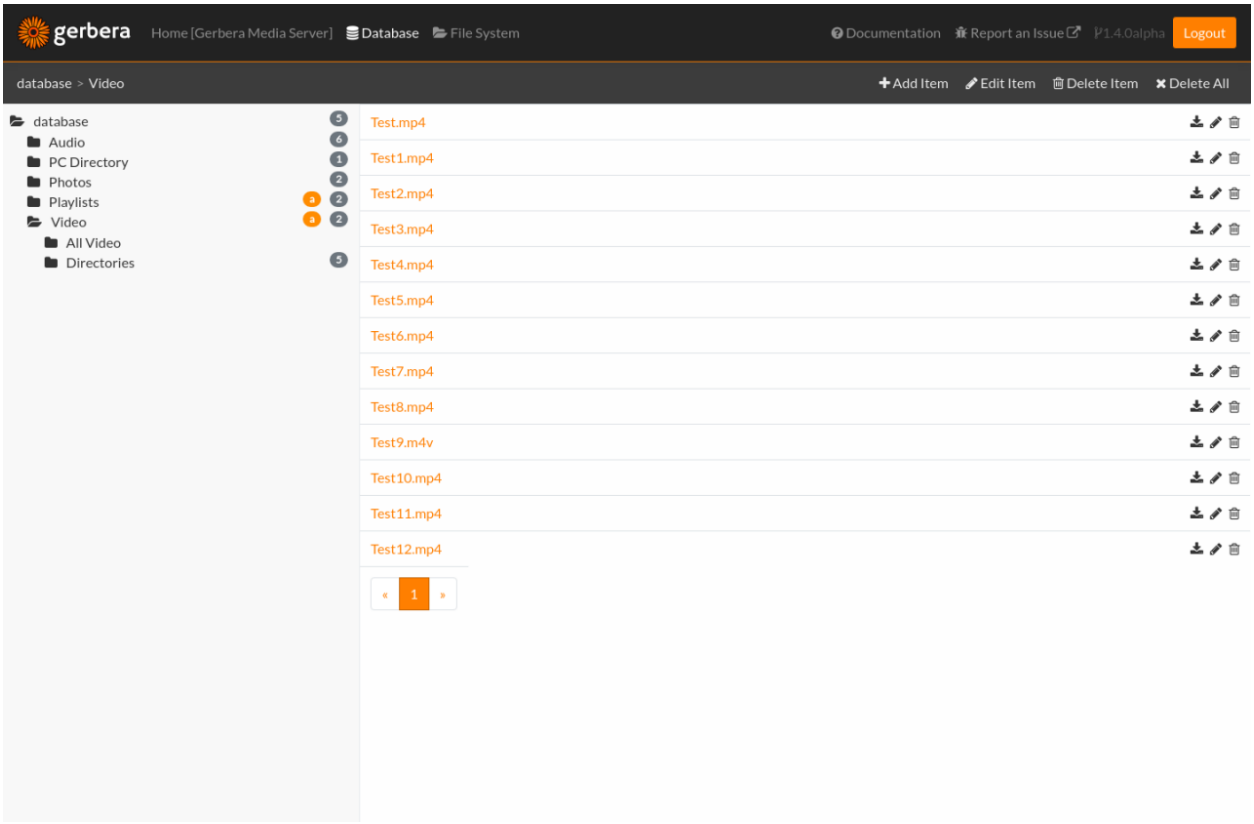
The filesystem view is accessible through the *File System* menu item. The filesystem view shows a folder tree representing the local filesystem. You can choose folders and items to add to your media library.



You can add items to the database view, making them accessible to UPNP clients. A successful addition of a filesystem item results in that item being indexed and available in the Database view.

Database View

The database view is accessible through the *Database* menu item. The view represents the virtual layout of your media library. The database view displays a tree structure, generated by the Gerbera *import* scripts. You can customize the database view structure by using the available scripts written using javascript.

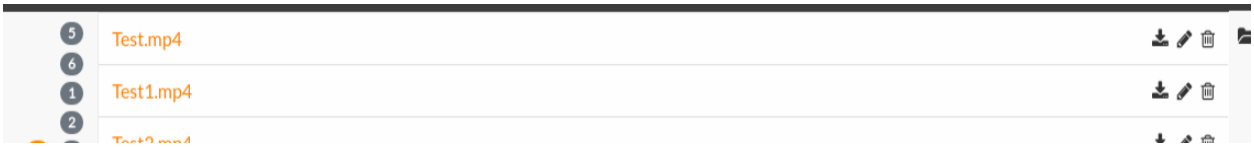


You can maintain the database view removing items and edit existing items to keep your media library up to date.

Item Operations

The items list displays when a virtual item is selected in the *database view* tree. The UI supports the following item operations

- Download Item
- Edit Item
- Delete Item



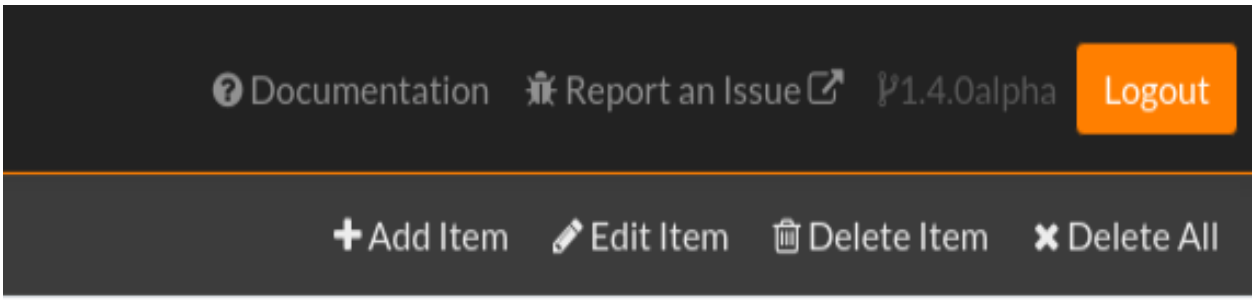
- Downloading the item retrieves the media directly from the Gerbera server.
- Editing the item updates the UPNP meta-data for the item.
- Deleting the item removes it from the virtual Database View

Item operations act upon existing database items. You can create new custom items using the Gerbera Trail.

Trail Operations

The Gerbera Trail shows the current database/filesystem path and provides a number of operations.

- Add New Item
- Add Autoscan
- Edit Container
- Delete Container
- Delete All



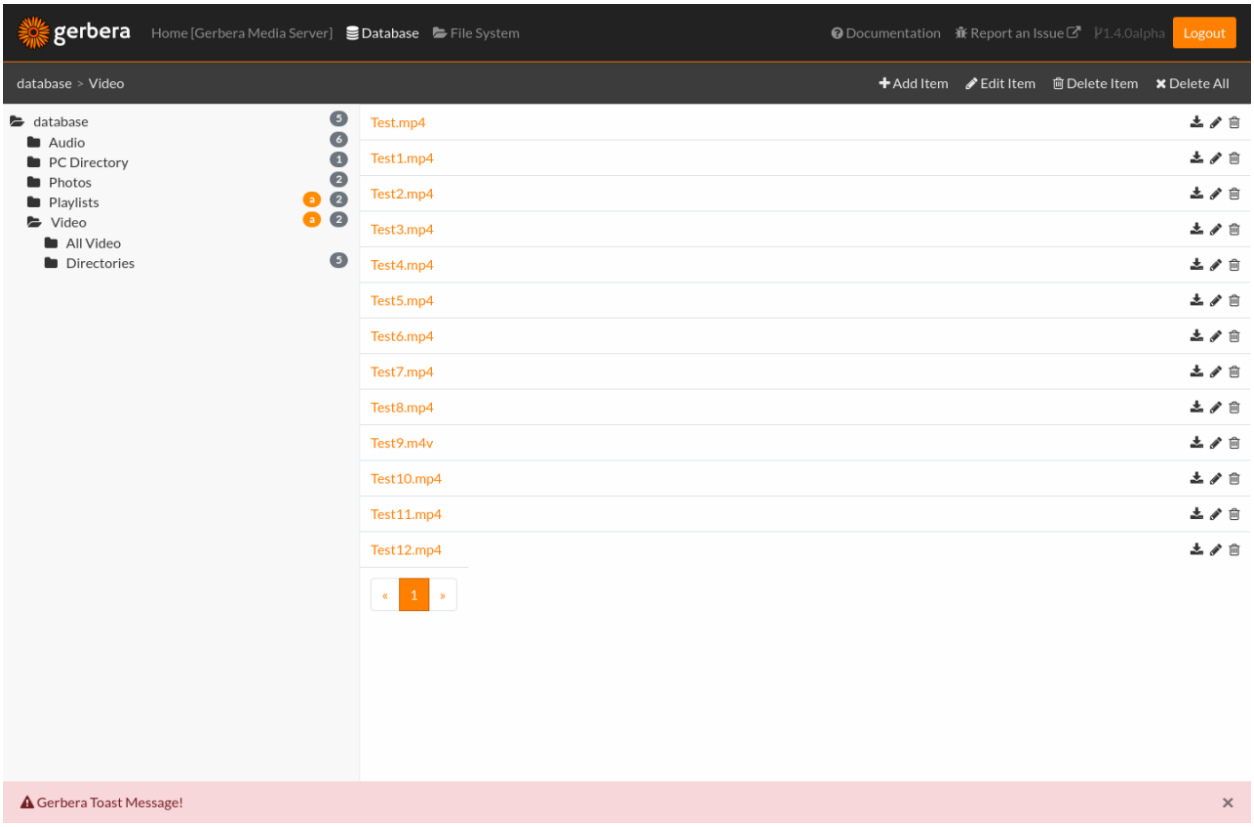
The Gerbera Trail supports adding virtual item types including containers, items, active items, internal urls, external urls. Adding an *autoscan* triggers the scan of a complete directory. Autoscan adds valid mime-type items to the Gerbera database. Edit and Delete of container removes the container. The *Delete All* icon removes a container and all sub-items from the Gerbera database.

Notifications

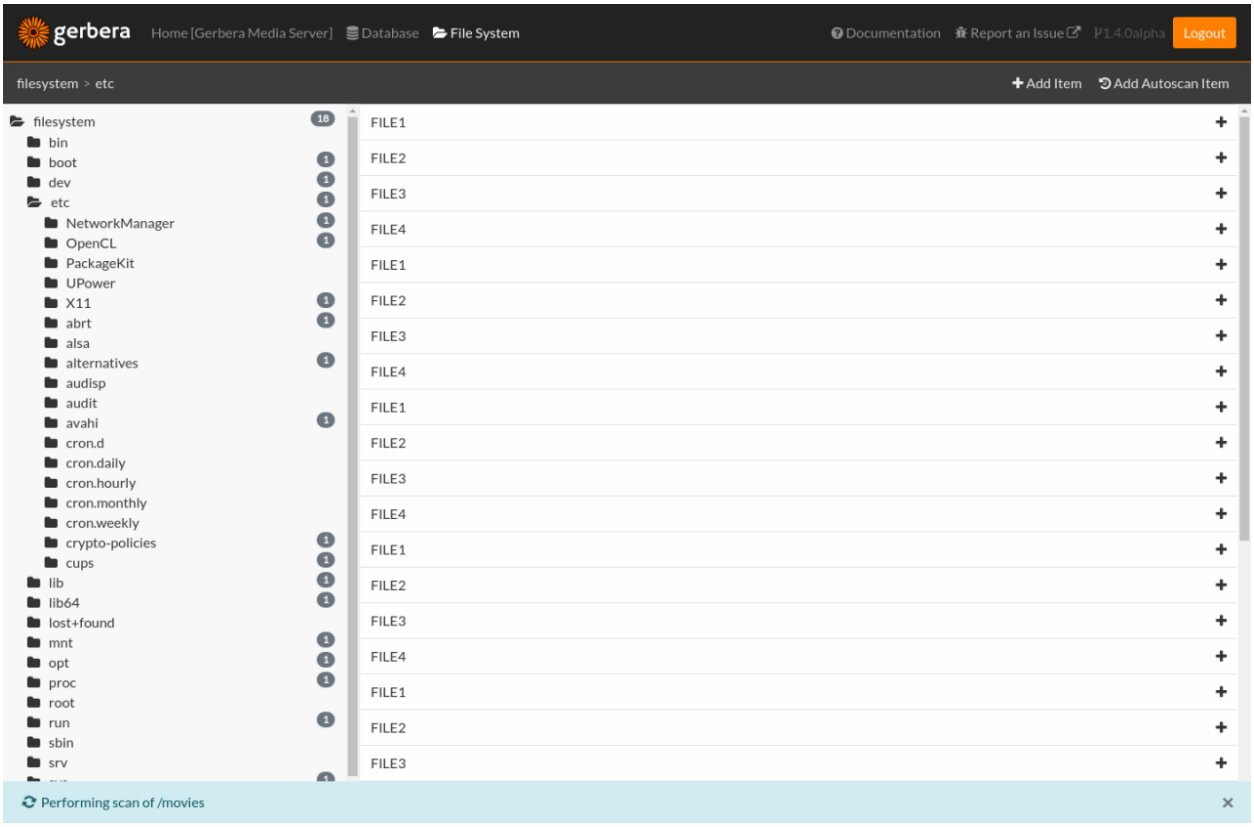
Gerbera runs all activities asynchronously. As you add content to the Gerbera database, the server scans and updates the items based on the media tagging. The Gerbera server reports busy activities to the UI in the form of 2 message bars.

- Status Message
- Task Message

The status message alerts upon operations to the server. The server reports success and failure messages to the status message bar at the bottom of the page.



The task message bar is only visible when there is active processing on the Gerbera server. The active tasks display in the task message bar at the top of the page.



4.2.5 Supported Devices

Attention Hardware Manufacturers:

If you want to improve compatibility between Gerbera and your renderer device or if you are interested in a port of Gerbera for your NAS device please contact Gerbera Contributors https://github.com/gerbera

MediaRenderers

Gerbera supports all UPnP compliant MediaRenderers, however there can always be various problems that depend on the particular device implementation. We always try to implement workarounds to compensate for failures and limitations of various renderers.

This is the list of client devices that Gerbera has been tested with and that are known to work. Please drop us a mail if you are using Gerbera with a device that is not in the list, report any success and failure. We will try to fix the issues and will add the device to the list.

Acer

- AT3705-MGW

Asus

- O!Play

Conceptronic

- C54WMP

Currys UK

- Logik IR100

Denon

- AVR-3808
- AVR-4306
- AVR-4308
- S-52
- ASD-3N

D-Link

- DSM-320
- DSM-320RD
- DSM-510
- DSM-520

Some additional settings in Gerbera configuration are required to enable special features for the DSM renderers. If you have a DSM-320 and are experiencing problems during AVI playback. Further, the DSM-320 behaves differently if it thinks that it is dealing with the D-Link server. Add the following to the server section of your configuration to enable srt subtitle support:

```
<manufacturerURL>redsonic.com</manufacturerURL>
<modelNumber>105</modelNumber>
```

It is still being investigated, but we were able to get subtitles working with a U.S. DSM-320 unit running firmware version 1.09

Also, the DSM-510 (probably also valid for other models) will only play avi files if the mimetype is set to video/avi, you may want to add a mapping for that to the extension-mimetype section in your config.xml:

```
<map from="avi" to="video/avi"/>
```

Freecom

- MusicPal

Häger

- OnAir (also known as BT Internet Radio)

HP

- MediaSmart TV

Users reported that after a firmwre upgrade the device stopped working properly. It seems that it does not sue the UPnP Browse action anymore, but now uses the optional Search action which is not implemented in Gerbera.

Hifidelio

- Hifidelio Pro-S

I-O Data

- AVeL LinkPlayer2 AVLP2/DVDLA

JVC

- DD-3
- DD-8

Kathrein

- UFS922

Kodak

- EasyShare EX-1011

Linn

- Sneaky DS

Linksys

- WMLS11B (Wireless-B Music System)
- KiSS 1600

Medion

- MD 85651

NeoDigits

- HELIOS X3000

Netgear

- EVA700
- MP101

Nokia

- N-95
- N-800

Odys

- i-net MusicBox

Philips

- Streamium SL-300i
- Streamium SL-400i
- Streamium MX-6000i
- Streamium NP1100
- Streamium MCi900
- WAS7500
- WAK3300
- WAC3500D
- SLA-5500
- SLA-5520
- 37PFL9603D

Pinnacle

- ShowCenter 200
- SoundBridge

Pioneer

- BDP-HD50-K
- BDP-94HD

Raidsonic

- IB-MP308HW-B

Revo

- Pico RadioStation

Roberts

- WM201 WiFi Radio

Playing OGG audio files requres a custom mimetype, add the following to the <extension-mimetype> section and reimport your OGGs:

```
<map from="ogg" to="audio/ogg"/>
```

Also, add this to the <mimetype-contenttype> section:

```
<treat mimetype="audio/ogg" as="ogg"/>
```

Roku

- SoundBridge M1001
- SoundBridge M2000

Sagem

- My Dual Radio 700

Siemens

- Gigaset M740AV

SMC

- EZ Stream SMCWAA-G

Snazio

- Snazio* Net DVD Cinema HD SZ1350

Sony

- Playstation 3

Firmware 1.80 introduces UPnP/DLNA support.

- Playstation 4

MediaPlayer seems to be flawed, so SSDP advertisements stop playback of videos. Set alive value in config.xml to e.g. 86400.

Syabas

- Popcorn Hour A110

T+A

- T+A Music Player

Tangent

- Quattro MkII

Telegent

- TG100

The TG100 client has a problem browsing containers, where item titles exceed 101 characters. We implemented a server-side workaround which allows you to limit the lengths of all titles and descriptions. Use the following settings in the <server> section of your configuration file:

```
<upnp-string-limit>101</upnp-string-limit>
```

TerraTec

- NOXON iRadio
- NOXON 2 Audio

Western Digital

- WD TV Live

Vistron

- MX-200I

Xtreamer

- Xtreamer

Yamaha

- RX-V2065

ZyXEL

- DMA-1000
- DMA-2500

Some users reported problems where the DMA will show an error ”Failed to retrieve list” and the DMA disconnecting from the server. Incresing the alive interval seems to solve the problem - add the following option to the <server> section of your configuration file:

```
<alive>600</alive>
```

Additionally, the DMA expects that avi files are serverd with the mime type of video/avi, so add the following to the <extensoin-mimetype> section in your configuration file:

```
<map from="avi" to="video/avi"/>
```

Also, add this to the <mimetype-contenttype> section:

```
<treat mimetype="video/avi" as="avi"/>
```

Network Attached Storage Devices

We provide a bitbake metadata file for the OpenEmbedded environment, it allows to easily cross compile Gerbera for various platforms. We have successfully tested Gerbera on ARM and MIPSel based devices, so it should be possible to install and run the server on various Linux based NAS products that are available on the market.

So far two devices are shipped with a preinstalled version of Gerbera, community firmware versions are available for the rest.

Asus

- WL500g

Use the statically linked mips32el binary package that is provided on our download site.

Buffalo

- KuroBox-HG
- LinkStation

Excito

- Bubba Mini Server (preinstalled)

Iomega

- StorCenter (preinstalled)

Linksys

- NSLU2

Available via Optware.

Maxtor

- MSS-I

Either use the Optware feeds or the statically linked mips2el binary package that is provided on our download site.

Raidsonic

- IB-NAS4200-B

Use the statically linked binary armv4 package that is provided on our download site.

Xtreamer

- Xtreamer eTRAYz

Western Digital

- MyBook

4.2.6 Transcoding Content

Transcoding allows you to perform format conversion of your content on the fly allowing you to view media that is otherwise not supported by your player.

For example, you might have your music collection stored in the OGG format, but your player only supports MP3 or you have your movies stored in DivX format, but your player only supports MPEG2 and MPEG4. Of course you could sit down and convert everything before viewing, but that is usually a time consuming procedure, besides, you often you want to keep your original data untouched and end up storing both, the converted and the original content - wasting space on your hard disk. That's where on the fly transcoding comes into play.

Another use case is online content - it is often presented in flv or asf formats, you may get mms or rtp streams which your player can not handle. The transcoding feature makes it possible to access such content.

Last but not least - subtitles. Only a few devices provide subtitle support, usually it's a proprietary solution not covered by UPnP. Using transcoding you can enable subtitles independent of the player device.

Theory of Operation

This chapter describes the idea behind the current transcoding implementation.

What Happens on the User Level

So how does this work? First, let's look at the normal situation where you are playing content that is natively supported by your player, let's say a DivX movie. You add it to the server, browse the content on your device, hit play and start streaming the content. Content that the player can not handle is usually grayed out in the on screen display or marked as unsupported.

Now, what happens if transcoding is in place?

First, you define transcoding profiles, specifying which formats should be converted, let's assume that you have some music stored in the FLAC format, but your device only supports MP3 and WAV. So, you can define that all FLAC media should be transcoded to WAV. You then start Gerbera and browse the content as usual on your device, if everything was set up correctly you should see that your FLAC files are marked as playable now. You hit play, just like usual, and you will see that your device starts playback.

Here is what happens in the background: when you browse Gerbera, we will look at the transcoding profile that you specified and, assuming the example above, tell your player that each FLAC file is actually a WAV file. Remember, we assumed that the player is capable of playing WAV content, so it will display the items as playable. As soon as you press play, we will use the options defined in the transcoding profile to launch the transcoder, we will feed it the original FLAC file and serve the transcoded WAV output directly to your player. The transcoding is done on the fly, the files are not stored on disk and do not require additional disk space.

Technical Background

The current implementation allows to plug in any application to do the transcoding. The only important thing is, that the application is capable of writing the output to a FIFO. Additionally, if the application is not capable of accessing online content directly we can proxy the online data and provide a FIFO for reading.

The application can be any executable and is launched as a process with a set of given parameters that are defined in the profile configuration. The special command line tokes %in and %out that are used in the profile will be substituted by the input file name or input URL and the output FIFO name.

So, the parameters tell the transcoding application: read content from this file, transcode it, and write the output to this FIFO. Gerbera will read the output from the FIFO and serve the transcoded stream to the player device.

Buffering is implemented to allow smooth playback and compensate for high bitrate scenes that may require more CPU power in the transcoding process.

Once you press stop or once you reach end of file we will make sure that the transcoding process is killed and we will clean up the FIFOs.

The chosen approach is extremely flexible and gives you maximum freedom of choice - you can also use this framework view mms and rtp streams even if this is originally not supported by your player, blend in subtitles or even listen to text documents using a text to speech processor.

> **Warning:** If using a wrapper script make sure that your script uses `exec` when calling the transcoder otherwise Gerbera will not be able to kill it.

Sample Configuration

We will not go through all possible configuration tags here, they are described in detail in the main documentation. Instead, we will show an sample configuration and describe the creation process.

First of all you need to decide what content has to be transcoded. It makes no sens to transcode something that can be played natively by your device. Next, you have to figure out how smart your device is - UPnP defines a way in

which it is possible to provide several resources (or several format representations) of the same content, however most devices only look at the first resource and ignore the rest. We implemented options to overcome this, however it may get tricky if you have several devices around and if each of them needs different settings.

All settings apply to your `config.xml`.

Profile Selection

What do we want to transcode? Let's assume that you have some .flv files on your drive or that you want to watch YouTube videos on your device using Gerbera. I have not yet heard of a UPnP player device that natively supports flash video, so let's tell Gerbera what we want to transcode all .flv content to something that our device understands.

This can be done in the mimetype-profile section under transcoding, mappings:

```
<transcode mimetype="video/x-flv" using="vlcprof"/>
```

So, we told Gerbera to transcode all video/x-flv content using the profile named `vlcprof`.

Profile Definition

We define `vlcprof` in the profiles section:

```
<profile name="vlcprof" enabled="yes" type="external">
    <mimetype>video/mpeg</mimetype>
    <agent command="vlc"
        arguments="-I dummy %in --sout #transcode{venc=ffmpeg,vcodec=mp2v,vb=4096,
↪fps=25,aenc=ffmpeg,acodec=mpga,ab=192,samplerate=44100,channels=2}:standard
↪{access=file,mux=ps,dst=%out} vlc:quit"/>
    <buffer size="10485760" chunk-size="131072" fill-size="2621440"/>
    <accept-url>yes</accept-url>
    <first-resource>yes</first-resource>
</profile>
```

Let's have a closer look:

```
<profile name="vlcprof" enabled="yes" type="external">
```

The profile tag defines the name of the profile - in our example it's `vlcprof`, it allows you to quickly switch the profile on and off by setting the enabled parameter to "yes" or "no" and also defines the profile type. Currently only one transcoding type is supported - "external".

Specifying The Target Mime Type

We need to define which mime type we are transcoding to - that's what the player device will see. It must be something it supports and there are also some other limitations: the output format must be streamable - meaning, it must be a format which can be played back without the need of seeking in the stream. AVI is a good example - it contains the index at the end of the file, so the player needs to seek (or use HTTP range requests) to read the index. Because of that you will not be able to transcode to AVI on the fly. A good target format is MPEG2 - it does not require the player to seek in the stream and it can be encoded on the fly with reasonable CPU power.

So, let's specify our target mime type:

```
<mimetype>video/mpeg</mimetype>
```

Bear in mind that this line only tells your player device about the content format, it does not tell anything to the transcoder application.

Choosing The Transcoder

Now it is time to look at the agent parameter - this tells us which application to execute and it also provides the necessary command line options for it:

```
<agent command="vlc" arguments="-I dummy %in --sout #transcode{venc=ffmpeg,
↪vcodec=mp2v,vb=4096,fps=25,aenc=ffmpeg,acodec=mpga,ab=192,samplerate=44100,
↪channels=2}:standard{access=file,mux=ps,dst=%out} vlc:quit"/>
```

In the above example the command to be executed is "vlc, it will be called with parameter specified in the arguments attribute. Note the special %in and %out tokens - they are not part of the vlc command line but have a special meaning in Gerbera. The %in token will be replaced by the input file name (i.e. the file that needs to be transcoded) and the %out token will be replaced by the output FIFO name, from where the transcoded content will be read by Gerbera and sent to the player.

Just to make it clearer:

```
<agent command="executable name" arguments="command line %in %out"/>
```

So, an agent tag defines the command which is an executable (make sure that it is in $PATH and that you have permissions to run it), and arguments which are the command line options and where %in and %out tokens are used in the place of the input and output file names.

Note: the output format produced by the transcoder must match the target mime type setting.

Buffer Settings

There are no defaults for the buffer settings, they need to be tuned to the performance of your system and also to the type of transcoded media if you want to achieve the best result.

The idea behind buffering is the following: let's assume that you are transcoding a high quality video, the source format has a variable bitrate. Your CPU can handle most scenes in real time, but occasionally some scenes have a higher bitrate which require more processing power. Without buffering you would not have a fluent playback - you would see stuttering during those high bitrate scenes. That's where buffering comes into play. Before sending the data to your player for the very first time, we will delay the start of the playback until the buffer is filled to a certain amount. This should give you enough slack to overcome those higher bitrate scenes and watch the movie without any stuttering or dropouts. Also, your CPU will not transcode the stream as fast as it is being played (i.e. real time), but work as fast as it can, filling up the buffer during lower bitrate scenes and thus giving you the chance to overcome even long scenes with high bitrate.

The buffer accepts three parameters and is defined like this:

```
<buffer size="5242880" chunk-size="102400" fill-size="1048576"/>
```

Size is the total size of the buffer, fill-size is the amount that has to be filled before sending out data from the buffer for the first time. Chunk-size is somewhat tricky, as you know we read the transcoded stream from a FIFO, we then put it into the buffer from where it gets served to the player. We read the data from the transcoder in chunks, once we fill up the chunk we put it into the buffer, so this setting is defining the size of those chunks. Lower values will make the buffer feel more responsive (i.e. it will be filled at a more fluent rate), however too low values will decrease performance. Also, do not set a too high value here since it may prevent smooth playback - data from the buffer is being played out, if you wait for a too big chunk at the same time you may empty the buffer.

Accepting Or Proxying Online Content

With Gerbera it is possible to add items that are not pointing to local content, but to online resources. It can be an mp3 stream, a YouTube video or some photos stored on the web. In case that the online media is stored in a format that is not supported by your player, you can use transcoding to convert it. Some transcoding applications, like VLC, handle online content pretty well, so you can give a URL directly to the transcoder and it will handle the data download itself. You can even use that to stream mms or rtsp streams, even if they are not directly supported by your player device. Some transcoders however, can not access online content directly but can only work with local data. For this situation we offer a special option:

```
<accept-url>no</accept-url>
```

If this option is set to "no" Gerbera will handle the download of the content and will feed the input to the transcoder via a FIFO. Of course the transcoding application must be capable of handling input from a FIFO. This only works for the HTTP protocol, we do not handle RTSP or MMS streams, use VLC is you want to handle those. When this option is set to "yes" we will give the URL to the transcoder.

Resource Index

What is a resource? In this case it's the <res> tag in the XML that is being sent to the player when it browses the server. Each item can have one or more resources, each resource describes the type of the content by specifying it's mime type and also tells the player how and where to get the content. So, resources within the item point to same content, but allow to present it in different formats. In case of transcoding we will offer the original data as well as the transcoded data by using the resource tags. A well implemented player will look at all resources that are available for the given item and choose the one that it supports. Unfortunately most players only look at the first resource and ignore the rest, this feature tells us to place the transcoded resource at the first position so that those renderers will see and take it.

```
<first-resource>yes</first-resource>
```

Hiding Original Resource

Sometimes it may be required that you only present the transcoded resource (read the previous section for explanation about resources) to the player. This option allows to do so:

```
<hide-original-resource>yes</hide-original-resource>
```

Advanced Settings

Sometimes you encounter a container format but want to transcode it only if it has a specific codec inside. Provided that Gerbera was compiled with ffmpeg support we offer fourcc based transcoding settings for AVI files. A sample configuration for a profile with fourcc specific settings would look like that:

```
<avi-fourcc-list mode="ignore">
    <fourcc>XVID</fourcc>
    <fourcc>DX50</fourcc>
</avi-fourcc-list>
```

Please refer to the main documentation on more information regarding the options.

We also provide a way to specify that a profile should only process the Theora codec if an OGG container is encountered:

```
<accept-ogg-theora>yes</accept-ogg-theora>
```

A new feature that was added in the 0.12 version possibility to specify that transcoded streams should be sent out using chunked HTTP encoding. This is now the default setting, since chunked encoding is preferred with content where the content length is not known. The setting can be controlled on a per profile basis using the following parameter:

```
<use-chunked-encoding>yes</use-chunked-encoding>
```

Testing And Troubleshooting

The external transcoding feature is very flexible, however there is a price for flexibility: a lot of things can go wrong. This section will try to cover the most common problems and present some methods on how things can be tested outside of Gerbera.

Testing the Transcoder

It's a good idea to test your transcoding application before putting together a profile. As described in the previous sections we get the transcoded stream via a FIFO, so it's important that the transcoder is capable of writing the output to a FIFO. This can be easily tested in the Linux command prompt.

Open a terminal and issue the following command:

```
mkfifo /tmp/tr-test
```

This will create a FIFO called tr-test in the /tmp directory. Open a second terminal, we will use one terminal to run the transcoder, and another one to examine the output.

For this test we will assume that we want to transcode an OGG file to WAV, the easiest way to do so is to use the ogg123 program which is part of the vorbis-tools package. Running ogg123 with the -d wav -f outfile parameter is exactly what we want, just remember that our outfile is the FIFO. So, run the following command, replacing some audio file with an OGG file that is available on your system, in one of the terminals:

```
ogg123 -d wav -f /tmp/tr-test /some/audio/file.ogg
```

The program will start and will appear to be hanging - it's blocked because noone is reading from the FIFO. While ogg123 is hanging, go to the second terminal and try playing directly from the FIFO (in this example we will use VLC to do that):

```
vlc /tmp/tr-test
```

If all goes well you should see that ogg123 is coming to life and you should hear the output from VLC - it should play the transcoded WAV stream.

Troubleshooting

This section will try to cover the most common problems related to the external transcoding feature.

Media Is Unplayable

What if the resulting stream is unplayable?

This can be the case with some media formats and contaeinrs. A good example is the AVI container - it contains the index at the very end of the file, meaning, that a player needs to seek to the end to get the index before rendering the video. Since seeking is not possible in transcoded streams you will not be able to transcode something to AVI and watch it from the FIFO.

Transcoding Does Not Start

As explained in the previous sections, transcoding only starts when your player issues an HTTP GET request to the server. Further, the request must be made to the transcoding URL.

Most common cases are:

- wrong mime type mapping: are you sure that you specified the source mime type correctly? Recheck the settings in the `<mimetype-profile>` section. If you are not sure about the source mime type of your media you can always check that via the web UI - just pick one of the files in question and click on the Edit icon.
- wrong output mime type: make sure that the mime type specified in the profile matches the media format that is produced by your transcoder.
- no permissions to execute the transcoding application: check that the user under which Gerbera is running has sufficient permissions to run the transcoding script or application.
- transcoding script is not executable or is not in `$PATH`: if you use a wrapper script around your transcoder, make sure that it is executable and can be found in `$PATH` (unless you specified an absolute name)

Problem Transcoding Online Streams

Some transcoding applications do not accept online content directly or have problems transcoding online media. If this is the case, set the `<accept-url>` option appropriately (currently Gerbera only supports proxying of HTTP streams). This will put the transcoder between two FIFOs, the online content will be downloaded by Gerbera and fed to the transcoder via a FIFO.

4.2.7 Scripting

Gerbera allows you to customize the structure of how your media is being presented to your renderer. One of the most important features introduced since the version 0.8 (*mediatomb*) are the virtual containers and virtual items.

Let's think of possible scenarios:

- You may want to separate your content by music, photo, video, maybe create a special container with all non playable stuff
- You may want your music to be sorted by genre, year, artist, album, or maybe by starting letters, so you can more easily find your favorite song when browsing the server
- You want to have your photos that you took with your favorite digital camera to appear in a special folder, or maybe you even want to separate the photos that you took with flash-on from the ones that you made without flash
- Your media player does not support video, so you do not even want to see the Video container
- It's up to your imagination :)

The scenarios described above and much more can be achieved with the help of an import script.

Gerbera supports a playlist parsing feature, which is also handled by scripting.

How It Works

This section will give you some overview on how virtual objects work and on how they are related to scripting.

Note: In order to use the import scripting feature you have to change the layout type from builtin to js in `config.xml`

Note: The sorting of Video and Photo items using the `rootpath` object is still somewhat experimental and not described here.

Understanding Virtual Objects

When you add a file or directory to the database via the web interface several things happen.

1. The object is inserted into the PC Directory. PC Directory is simply a special non-removable container. Any media file added will have an entry inside the PC Directory tree. PC Directory's hierarchy reflects the file system hierarchy, all objects inside the PC Directory including itself are NON-VIRTUAL objects. All virtual objects may have a different title, description, etc., but they are still references to objects in the PC-Directory. That's why it is not possible to change a location of a virtual object - the only exceptions are URL items and Active items.
2. Once an item is added to the PC Directory it is forwarded to the virtual object engine. The virtual object engine's mission is to organize and present the media database in a logical hierarchy based on the available metadata of the items.

Each UPnP server implements this so called virtual object hierarchy in a different way. Audio files are usually sorted by artist, album, some servers may just present a view similar to the file system and so on. Most servers have strong limitations on the structure of the virtual containers, they usually offer a predefined layout of data and the user has to live with it. In Gerbera we try to address this shortcoming by introducing the scriptable virtual object engine. It is designed to be:

- maximally flexible
- easily customizable and extendable
- robust and efficient

We try to achieve these goals by embedding a scripting runtime environment that allows the execution of ECMAScript E5/5.1 conform scripts better known as JavaScript. Gerbera uses duktape scripting engine to run JavaScript.

Theory of Operation

After an item is added to the PC Directory it is automatically fed as input to the import script. The script then creates one or more virtual items for the given original item. Items created from scripts are always marked virtual.

When the virtual object engine gets notified of an added item, following happens: a javascript object is created mirroring the properties of the item. The object is introduced to the script environment and bound to the predefined variable 'orig'. This way a variable orig is always defined for every script invocation and represents the original data of the added item. Then the script is invoked.

Note: In the current implementation, if you modify the script then you will have to restart the server for the new logic to take ffect.

The script is only triggered when new objects are added to the database, also note that the script does not modify any objects that already exist in the database - it only processes new objects that are being added.

When a playlist item is encountered, it is automatically fed as input to the playlist script. The playlist script attempts to parse the playlist and adds new item to the database, the item is then processed by the import script.

Global Variables And Constants

In this section we will introduce the properties of the object that will be processed by the script, as well as functions that are offered by the server.

The Media Object

Each time an item is added to the database the import script is invoked. So, one script invocation processes exactly one non virtual item, and creates a number of virtual items and containers. The original item is made available in the form of the global variable 'orig'. Additionally, when the object being imported is a playlist, it is made available to the playlist parser script in the form of the global variable 'playlist'. It is usually a good idea to only read from these variables and to create and only modify local copies.

Note: modifying the properties of the orig object will not propagate the changes to the database, only a call to the `addCdsObject()` will permanently add the object.

General Properties

Here is a list of properties of an object, you can set them you create a new object or when you modify a copy of the 'orig' object.

RW means read/write, i.e. - changes made to that property will be transferred into the database.

RO means, that this is a read only property, any changes made to it will get lost.

`orig.`**`objectType`**
RW

This defines the object type, following types are available:

Key	Description
OBJECT_TYPE_CONTAINER	Object is a container
OBJECT_TYPE_ITEM	Object is an item
OBJECT_TYPE_ACTIVE_ITEM	Object is an active item
OBJECT_TYPE_ITEM_EXTERNAL_URL	Object is a link to a resource on the Internet
OBJECT_TYPE_ITEM_INTERNAL_URL	Object is an internal link

`orig.`**`title`**
RW

This is the title of the original object, since the object represents an entry in the PC-Directory, the title will be set to it's file name. This field corresponds to dc:title in the DIDL-Lite XML.

`orig.`**`id`**
RO

The object ID, make sure to set all refID's (reference IDs) of your virtual objects to that ID.

`orig.`**`parentID`**
RO

The object ID of the parent container.

`orig.`**`upnpclass`**
RW

The UPnP class of the item, this corresponds to `upnp:class` in the DIDL-Lite XML.

`orig.`**`location`**
RO

Location on disk, given by the absolute path and file name.

`orig.`**`theora`**
RO

This property is a boolean value, it is non zero if the particular item is of type OGG Theora. This is useful to allow proper sorting of media and thus placing OGG Vorbis into the Audio container and OGG Theora into the Video container.

`orig.`**`onlineservice`**
RO

Identifies if the item belongs to an online service and thus has extended properties. Following types are available:

Key	Description
ONLINE_SERVICE_NONE	The item does not belong to an online service and does not have extended properties.
ONLINE_SERVICE_APPLE_TRAILERS	The item belongs to the Apple Trailers service and has extended properties.

`orig.`**`mimetype`**
RW

Mimetype of the object.

`orig.`**`meta`**
RW

Array holding the metadata that was extracted from the object (i.e. id3/exif/etc. information)

`orig.orig.`**`meta[M_TITLE]`**
RW

Extracted title (for example the id3 title if the object is an mp3 file), if you want that your new virtual object is displayed under this title you will have to set *obj.title = orig.meta[M_TITLE]*

`orig.`**`meta[M_ARTIST]`**
RW

Artist information, this corresponds to `upnp:artist` in the DIDL-Lite XML.

`orig.`**`meta[M_ALBUM]`**
RW

Album information, this corresponds to `upnp:album` in the DIDL-Lite XML.

`orig.`**`meta[M_DATE]`**
RW

Date, must be in the format of **YYYY-MM-DD** (required by the UPnP spec), this corresponds to dc:date in the DIDL-Lite XML.

`orig.`**`meta[M_GENRE]`**
RW

Genre of the item, this corresponds to `upnp:genre` in the DIDL-Lite XML.

`orig.`**`meta[M_DESCRIPTION]`**
RW

Description of the item, this corresponds to `dc:description` in the DIDL-Lite XML.

`orig.`**`meta[M_REGION]`**
RW

Region description of the item, this corresponds to `upnp:region` in the DIDL-Lite XML.

`orig.`**`meta[M_TRACKNUMBER]`**
RW

Track number of the item, this corresponds to `upnp:originalTrackNumber` in the DIDL-Lite XML.

`orig.`**`meta[M_AUTHOR]`**
RW

Author of the media, this corresponds to `upnp:author` in the DIDL-Lite XML.

`orig.`**`meta[M_DIRECTOR]`**
RW

Director of the media, this corresponds to `upnp:director` in the DIDL-Lite XML.

`orig.`**`meta[M_PUBLISHER]`**
RW

Director of the media, this corresponds to `dc:publisher` in the DIDL-Lite XML.

`orig.`**`meta[M_RATING]`**
RW

Director of the media, this corresponds to `upnp:rating` in the DIDL-Lite XML.

`orig.`**`meta[M_ACTOR]`**
RW

Director of the media, this corresponds to `upnp:actor` in the DIDL-Lite XML.

`orig.`**`meta[M_PRODUCER]`**
RW

Director of the media, this corresponds to `upnp:producer` in the DIDL-Lite XML.

`orig.`**`aux`**
RO

Array holding the so called auxiliary data. Aux data is metadata that is not part of UPnP, for example - this can be a camera model that was used to make a photo, or the information if the photo was taken with or without flash.

Currently aux data can be gathered from **libexif** (see the Import section in the main documentation for more details). So, this array will hold the tags that you specified in your config.xml, allowing you to create your virtual structure according to your liking.

orig.**playlistOrder**

RW

This property is only available if the object is being created by the playlist script. It's similar to ID3 track number, but is used to set the position of the newly created object inside a parsed playlist container. Usually you will increment the number for each new object that you create while parsing the playlist, thus ensuring that the resulting order is the same as in the original playlist.

Constants

Actually there are no such things as constants in JS, so those are actually predefined global variables that are set during JS engine initialization. Do not assign any values to them, otherwise following script invocation will be using wrong values.

Constant	Type	Value	Notes
UPNP_CLASS_CONTAINER	string	object.container	
UPNP_CLASS_CONTAINER_MUSIC_ARTIST	string	object.container.person.musicArtist	
UPNP_CLASS_CONTAINER_MUSIC_GENRE	string	object.container.genre.musicGenre	
UPNP_CLASS_CONTAINER_MUSIC_ALBUM	string	object.container.album.musicAlbum	This container class will be treated by the server in a special way, all music items in this container will be sorted by ID3 track number.
UPNP_CLASS_PLAYLIST_CONTAINER	string	object.container.playlistContainer	This container class will be treated by the server in a special way, all items in this container will be sorted by the number specified in the playlistOrder property (this is set when an object is created by the playlist script).
UPNP_CLASS_ITEM	string	object.item	
UPNP_CLASS_ITEM_MUSIC_TRACK	string	object.item.audioItem.musicTrack	
UPNP_CLASS_ITEM_VIDEO	string	object.item.videoItem	
UPNP_CLASS_ITEM_IMAGE	string	object.item.imageItem	
OBJECT_TYPE_CONTAINER	integer	1	
OBJECT_TYPE_ITEM	integer	2	
OBJECT_TYPE_ACTIVE_ITEM	integer	4	
OBJECT_TYPE_ITEM_EXTERNAL_URL	integer	8	
OBJECT_TYPE_ITEM_INTERNAL_URL	integer	16	

Functions

The server offers various native functions that can be called from the scripts, additionally there are some js helper functions that can be used.

Native Server Functions

The so called native functions are implemented in C++ in the server and can be called from the scripts.

Native Functions Available To All Scripts

The server offers three functions which can be called from within the import and/or the playlist script:

addCdsObject (*object*, *containerChain*, *lastContainerClass*)
Adds the object as a virtual object to the container chain

Arguments

- **object** (*object*) – A virtual object that is either a copy of or a reference to 'orig'
- **containerChain** (*string*) – A string, defining where the object will be added in the database hierarchy. The containers in the chain are separated by a slash '/', for example, a value of '/Audio/All Music' will add the object to the Audio, All Music container in the server hierarchy. Make sure to properly escape the slash characters in container names. You will find more information on container chain escaping later in this chapter.
- **lastContainerClass** (*string*) – A string, defining the upnp:class of the container that appears last in the chain. This parameter can be omitted, in this case the default value `object.container` will be taken. Setting specific upnp container classes is useful to define the special meaning of a particular container; for example, the server will always sort songs by track number if upnp class of a container is set to `object.container.album.musicAlbum`.

copyObject (*originalObject*)
This function returns a copy of the virtual object.

Arguments

- **originalObject** (*object*) –

Returns A copy of the virtual object

print (...)
This function is useful for debugging scripts, it simply prints to the standard output.

f2i (*string*)
Converts filesystem charset to internal UTF-8.

Arguments

- **string** (*string*) –

The 'from' charsets can be defined in the server configuration

m2i (*string*)
Converts metadata charset to internal UTF-8.

Arguments

- **string** (*string*) –

The 'from' charsets can be defined in the server configuration

p2i(*string*)
Converts playlist charset to internal UTF-8.

Arguments

- **string**(*string*) –

The 'from' charsets can be defined in the server configuration

j2i(*string*)
Converts js charset to internal UTF-8.

Arguments

- **string**(*string*) –

The 'from' charsets can be defined in the server configuration

Native Functions Available To The Playlist Script

The following function is only available to the playlist script.

readln()

Returns string

This function reads and returns exactly one line of text from the playlist that is currently being processed, end of line is identified by carriage return/line feed characters. Each subsequent call will return the next line, there is no way to go back. The idea is, that you can process your playlist line by line and gather the required information to create new objects which can be added to the database.

Helper Functions

There is a set of helper JavaScript functions which reside in the common.js script. They can be used by the import and by the playlist script.

escapeSlash(*name*)
Escapes slash '/' characters in a string. This is necessary, because the container chain is defined by a slash separated string, where slash has a special meaning - it defines the container hierarchy. That means, that slashes that appear in the object's title need to be properly escaped.

Arguments

- **name** (*string*) – A string to be escaped

Returns string

createContainerChain(*arr*)
Verifies that the names are properly escaped and adds the slash separators as necessary

Arguments

- **arr** (*array*) – An array of container names

Returns string formatted for use in `addCdsObject`

```
function createContainerChain(arr)
{
    var path = '';
    for (var i = 0; i < arr.length; i++)
    {
        path = path + '/' + escapeSlash(arr[i]);
    }
    return path;
}
```

getYear (*date*)

Arguments

- **string** – A date formatted in `yyyy-mm-dd`

Returns string - Year value

```
function getYear(date)
{
    var matches = date.match(/^([0-9]{4})-/);
    if (matches)
        return matches[1];
    else
        return date;
}
```

getPlaylistType (*mimetype*)

This function identifies the type of the playlist by the mimetype, it is used in the playlist script to select an appropriate parser.

Arguments

- **string** – A valid mime-type

Returns string - playlist type

```
function getPlaylistType(mimetype)
{
    if (mimetype == 'audio/x-mpegurl')
        return 'm3u';
    if (mimetype == 'audio/x-scpls')
        return 'pls';
    return '';
}
```

Walkthrough

Now it is time to take a closer look at the default scripts that are supplied with Gerbera. Usually it is installed in the `/usr/share/gerbera/js/` directory, but you will also find it in `scripts/js/` in the Gerbera source tree.

Note: this is not a JavaScript tutorial, if you are new to JS you should probably make yourself familiar with the language.

Import Script

We start with a walkthrough of the default import script, it is called import.js in the Gerbera distribution.

Below are the import script functions that organize our content in the database by creating the virtual structure. Each media type - audio, image and video is handled by a separate function.

Audio Content Handler

The biggest one is the function that handles audio - the reason is simple: mp3 files offer a lot of metadata like album, artist, genre, etc. information, this allows us to create a nice container layout.

```
function addAudio(obj) {
    var desc = '';
    var artist_full;
    var album_full;

    // First we will gather all the metadata that is provided by our
    // object, of course it is possible that some fields are empty -
    // we will have to check that to make sure that we handle this
    // case correctly.
    var title = obj.meta[M_TITLE];

    // Note the difference between obj.title and obj.meta[M_TITLE] -
    // while object.title will originally be set to the file name,
    // obj.meta[M_TITLE] will contain the parsed title - in this
    // particular example the ID3 title of an MP3.
    if (!title) {
        title = obj.title;
    }

    var artist = obj.meta[M_ARTIST];
    if (!artist) {
        artist = 'Unknown';
        artist_full = null;
    } else {
        artist_full = artist;
        desc = artist;
    }

    var album = obj.meta[M_ALBUM];
    if (!album) {
        album = 'Unknown';
        album_full = null;
    } else {
        desc = desc + ', ' + album;
        album_full = album;
    }

    if (desc) {
        desc = desc + ', ';
    }
    desc = desc + title;

    var date = obj.meta[M_DATE];
    if (!date) {
```

(continues on next page)

(continued from previous page)

```
        date = 'Unknown';
    } else {
        date = getYear(date);
        obj.meta[M_UPNP_DATE] = date;
        desc = desc + ', ' + date;
    }

    var genre = obj.meta[M_GENRE];
    if (!genre) {
        genre = 'Unknown';
    } else {
        desc = desc + ', ' + genre;
    }

    var description = obj.meta[M_DESCRIPTION];
    if (!description) {
        obj.description = desc;
    }

    var composer = obj.meta[M_COMPOSER];
    if (!composer) {
        composer = 'None';
    }

    var conductor = obj.meta[M_CONDUCTOR];
    if (!conductor) {
        conductor = 'None';
    }

    var orchestra = obj.meta[M_ORCHESTRA];
    if (!orchestra) {
        orchestra = 'None';
    }

    // uncomment this if you want to have track numbers in front of the title
    // in album view
/*
    var track = obj.meta[M_TRACKNUMBER];
    if (!track) {
        track = '';
    } else {
        if (track.length == 1) {
            track = '0' + track;
        }
        track = track + ' ';
    }
*/
    // comment the following line out if you uncomment the stuff above  :)
    var track = '';

// uncomment this if you want to have channel numbers in front of the title
/*
    var channels = obj.res[R_NRAUDIOCHANNELS];
    if (channels) {
        if (channels === "1") {
            track = track + '[MONO]';
        } else if (channels === "2") {
```

(continues on next page)

(continued from previous page)

```
            track = track + '[STEREO]';
        } else {
            track = track + '[MULTI]';
        }
    }
*/

    // We finally gathered all data that we need, so let's create a
    // nice layout for our audio files. Note how we are constructing
    // the chain, in the line below the array 'chain' will be
    // converted to 'Audio/All audio' by the createContainerChain()
    // function.

    var chain = ['Audio', 'All Audio'];
    obj.title = title;

    // The UPnP class argument to addCdsObject() is optional, if it is
    // not supplied the default UPnP class will be used. However, it
    // is suggested to correctly set UPnP classes of containers and
    // objects - this information may be used by some renderers to
    // identify the type of the container and present the content in a
    // different manner .

    addCdsObject(obj, createContainerChain(chain));

    chain = ['Audio', 'Artists', artist, 'All Songs'];
    addCdsObject(obj, createContainerChain(chain));

    chain = ['Audio', 'All - full name'];
    var temp = '';
    if (artist_full) {
        temp = artist_full;
    }

    if (album_full) {
        temp = temp + ' - ' + album_full + ' - ';
    } else {
        temp = temp + ' - ';
    }

    obj.title = temp + title;
    addCdsObject(obj, createContainerChain(chain));

    chain = ['Audio', 'Artists', artist, 'All - full name'];
    addCdsObject(obj, createContainerChain(chain));

    chain = ['Audio', 'Artists', artist, album];
    obj.title = track + title;
    addCdsObject(obj, createContainerChain(chain), UPNP_CLASS_CONTAINER_MUSIC_ALBUM);

    chain = ['Audio', 'Albums', album];
    obj.title = track + title;

    // Remember, the server will sort all items by ID3 track if the
    // container class is set to UPNP_CLASS_CONTAINER_MUSIC_ALBUM.

    addCdsObject(obj, createContainerChain(chain), UPNP_CLASS_CONTAINER_MUSIC_ALBUM);
```

(continues on next page)

(continued from previous page)

```
    chain = ['Audio', 'Genres', genre];
    addCdsObject(obj, createContainerChain(chain), UPNP_CLASS_CONTAINER_MUSIC_GENRE);

    chain = ['Audio', 'Year', date];
    addCdsObject(obj, createContainerChain(chain));

    chain = ['Audio', 'Composers', composer];
    addCdsObject(obj, createContainerChain(chain), UPNP_CLASS_CONTAINER_MUSIC_
↪COMPOSER);
}
```

Image Content Handler

This function takes care of images. Currently it does very little sorting, but could easily be extended - photos made by digital cameras provide lots of information in the Exif tag, so you could easily add code to sort your pictures by camera model or anything Exif field you might be interested in.

Note: if you want to use those additional Exif fields you need to compile MediaTomb with libexif support and also specify the fields of interest in the import section of your configuration file (See documentation about library-options).

```
function addImage(obj) {
    var chain = ['Photos', 'All Photos'];
    addCdsObject(obj, createContainerChain(chain), UPNP_CLASS_CONTAINER);

    var date = obj.meta[M_DATE];
    if (date) {
        var dateParts = date.split('-');
        if (dateParts.length > 1) {
            var year = dateParts[0];
            var month = dateParts[1];

            chain = ['Photos', 'Year', year, month];
            addCdsObject(obj, createContainerChain(chain), UPNP_CLASS_CONTAINER);
        }

        chain = ['Photos', 'Date', date];
        addCdsObject(obj, createContainerChain(chain), UPNP_CLASS_CONTAINER);
    }

    var dir = getRootPath(object_script_path, obj.location);

    if (dir.length > 0) {
        chain = ['Photos', 'Directories'];
        chain = chain.concat(dir);
        addCdsObject(obj, createContainerChain(chain));
    }
}
```

Just like in the addAudio() function - we construct our container chain and add the object.

Video Content Handler

Not much to say here… I think libextractor is capable of retrieving some information from video files, however I seldom encountered any video files populated with metadata. You could also try ffmpeg to get more information, however by default we keep it very simple - we just put everything into the 'All Video' container.

```
function addVideo(obj) {
    var chain = ['Video', 'All Video'];
    addCdsObject(obj, createContainerChain(chain));

    var dir = getRootPath(object_script_path, obj.location);

    if (dir.length > 0) {
        chain = ['Video', 'Directories'];
        chain = chain.concat(dir);
        addCdsObject(obj, createContainerChain(chain));
    }
}
```

Apple Trailers Content Handler

This function processes items that are importent via the Apple Trailers feature. We will organize the trailers by genre, post date and release date, additionally we will also add a container holding all trailers.

```
function addTrailer(obj) {
    var chain;

    // First we will add the item to the 'All Trailers' container, so
    // that we get a nice long playlist:

    chain = ['Online Services', 'Apple Trailers', 'All Trailers'];
    addCdsObject(obj, createContainerChain(chain));

    // We also want to sort the trailers by genre, however we need to
    // take some extra care here: the genre property here is a comma
    // separated value list, so one trailer can have several matching
    // genres that will be returned as one string. We will split that
    // string and create individual genre containers.

    var genre = obj.meta[M_GENRE];
    if (genre) {

        // A genre string "Science Fiction, Thriller" will be split to
        // "Science Fiction" and "Thriller" respectively.

        genres = genre.split(', ');
        for (var i = 0; i < genres.length; i++) {
            chain = ['Online Services', 'Apple Trailers', 'Genres', genres[i]];
            addCdsObject(obj, createContainerChain(chain));
        }
    }

    // The release date is offered in a YYYY-MM-DD format, we won't do
    // too much extra checking regading validity, however we only want
    // to group the trailers by year and month:
```

(continues on next page)

(continued from previous page)

```
    var reldate = obj.meta[M_DATE];
    if ((reldate) && (reldate.length >= 7)) {
        chain = ['Online Services', 'Apple Trailers', 'Release Date', reldate.slice(0,
↪ 7)];
        addCdsObject(obj, createContainerChain(chain));
    }

    // We also want to group the trailers by the date when they were
    // originally posted, the post date is available via the aux
    // array. Similar to the release date, we will cut off the day and
    // create our containres in the YYYY-MM format.

    var postdate = obj.aux[APPLE_TRAILERS_AUXDATA_POST_DATE];
    if ((postdate) && (postdate.length >= 7)) {
        chain = ['Online Services', 'Apple Trailers', 'Post Date', postdate.slice(0,␣
↪7)];
        addCdsObject(obj, createContainerChain(chain));
    }
}
```

Putting it all together

This is the main part of the script, it looks at the mimetype of the original object and feeds the object to the appropriate content handler.

```
if (getPlaylistType(orig.mimetype) === '') {
    var arr = orig.mimetype.split('/');
    var mime = arr[0];

    // All virtual objects are references to objects in the
    // PC-Directory, so make sure to correctly set the reference ID!

    var obj = orig;
    obj.refID = orig.id;

    if (mime === 'audio') {
        addAudio(obj);
    }

    if (mime === 'video') {
        if (obj.onlineservice === ONLINE_SERVICE_APPLE_TRAILERS) {
            addTrailer(obj);
        } else {
            addVideo(obj);
        }
    }

    if (mime === 'image') {
        addImage(obj);
    }

    // We now also have OGG Theora recognition, so we can ensure that
    // Vorbis
    if (orig.mimetype === 'application/ogg') {
```

(continues on next page)

(continued from previous page)

```
        if (orig.theora === 1) {
            addVideo(obj);
        } else {
            addAudio(obj);
        }
    }
}
```

Playlist Script

The default playlist parsing script is called playlists.js, similar to the import script it works with a global object which is called 'playlist', the fields are similar to the 'orig' that is used in the import script with the exception of the playlistOrder field which is special to playlists.

Another big difference between playlist and import scripts is, that playlist scripts can add new media to the database, while import scripts only process already existing objects (the ones found in PC Directory) and just add additional virtual items.

The default playlist script implementation supports parsing of m3u and pls formats, but you can add support for parsing of any ASCII based playlist format.

Adding Items

We will first look at a helper function:

```
addPlaylistItem(location, title, playlistChain);
```

It is defined in playlists.js, it receives the location (path on disk or HTTP URL), the title and the desired position of the item in the database layout (remember the container chains used in the import script).

The function first decides if we are dealing with an item that represents a resource on the web, or if we are dealing with a local file. After that it populates all item fields accordingly and calls the addCdsObject() that was introduced earlier. Note, that if the object that is being added by the playlist script is not yet in the database, the import script will be invoked.

Below is the complete function with some comments:

```
function addPlaylistItem(location, title, playlistChain, order) {
    // Determine if the item that we got is an URL or a local file.

    if (location.match(/^.*:\/\//)) {
        var exturl = {};

        // Setting the mimetype is crucial and tricky... if you get it
        // wrong your renderer may show the item as unsupported and refuse
        // to play it. Unfortunately most playlist formats do not provide
        // any mimetype information.
        exturl.mimetype = 'audio/mpeg';

        // Make sure to correctly set the object type, then populate the
        // remaining fields.
        exturl.objectType = OBJECT_TYPE_ITEM_EXTERNAL_URL;

        exturl.location = location;
        exturl.title = (title ? title : location);
```

(continues on next page)

(continued from previous page)

```
        exturl.protocol = 'http-get';
        exturl.upnpclass = UPNP_CLASS_ITEM_MUSIC_TRACK;
        exturl.description = "Song from " + playlist.title;

        // This is a special field which ensures that your playlist files
        // will be displayed in the correct order inside a playlist
        // container. It is similar to the id3 track number that is used
        // to sort the media in album containers.
        exturl.playlistOrder = (order ? order : playlistOrder++);

        // Your item will be added to the container named by the playlist
        // that you are currently parsing.
        addCdsObject(exturl, playlistChain,  UPNP_CLASS_PLAYLIST_CONTAINER);
    } else {
        if (location.substr(0,1) !== '/') {
            location = playlistLocation + location;
        }
        var cds = getCdsObject(location);
        if (!cds) {
            print("Skipping item: " + location);
            return
        }

        var item = copyObject(cds);

        item.playlistOrder = (order ? order : playlistOrder++);
        item.title = item.meta[M_TITLE];

        addCdsObject(item, playlistChain,  UPNP_CLASS_PLAYLIST_CONTAINER);
    }
}
```

Main Parsing

The actual parsing is done in the main part of the script. First, the type of the playlist is determined (based on the playlist mimetype), then the correct parser is chosen. The parsing itself is a loop, where each call to readln() returns exactly one line of text from the playlist. There is no possibility to go back, each readln() invocation will retrieve the next line until end of file is reached.

To keep things easy we will only list the m3u parsing here. Again, if you are not familiar with regular expressions, now is probably the time to take a closer look.

```
else if (type === 'm3u') {
    title = null;
    line = readln();

    // Here is the do - while loop which will read the playlist line by line.
    do {
        var matches = line.match(/^#EXTINF:(-?\d+),\s?(\S.+)$/i);
        if (matches) {
            // duration = matches[1]; // currently unused
            title = matches[2];
        }
        else if (!line.match(/^(#|\s*$)/)) {
            // Call the helper function to add the item once you gathered the data:
```

(continues on next page)

(continued from previous page)

```
            addPlaylistItem(line, title, playlistChain);

            // Also add to "Directories"
            if (playlistDirChain)
                addPlaylistItem(line, title, playlistDirChain);

            title = null;
        }

        line = readln();
    } while (line);
}
```

Happy scripting!

4.2.8 Configuration

Gerbera is highly configurable and allows the user to set various options and preferences that define the server's behavior. Rather than enforcing certain features upon the user, we prefer to offer a number of choices where possible. The heart of Gerbera configuration is the config.xml file, which is located in the `~/.config/gerbera` directory. If the configuration file is not found in the default location and no configuration was specified on the command line, Gerbera will generate a default config.xml file in the `~/.config/gerbera` directory. The file is in the XML format and can be edited by a simple text editor, here is the list of all available options:

- **Required** means that the server will not start if the tag is missing in the configuration.
- **Optional** means that the tag can be left out of the configuration file.

The root tag of Gerbera configuration is:

```
<?xml version="1.0" encoding="UTF-8"?>
<config version="2"
    xmlns="http://mediatomb.cc/config/2"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://mediatomb.cc/config/2 http://mediatomb.cc/config/2.xsd
↪">
    ...
</config>
```

The server configuration file has several options. Below are links to the configuration sections

- *Server*
- *Extended Runtime Options*
- *Import Content*
- *Online Content*
- *Transcode Content*

Generating Configuration

A Gerbera instance requires a configuration file to launch.

Gerbera provides a command line flag `--create-config` to generate a default `config.xml` file. You will need to generate the configuration file when running Gerbera for the first time. Gerbera reports the missing configuration upon startup similar to the message below

```
The server configuration file could not be found in ~/.config/gerbera
Gerbera could not find a default configuration file.
Try specifying an alternative configuration file on the command line.
For a list of options run: gerbera -h
```

- Run command to create configuration

```
$ gerbera --create-config
```

This command outputs the `config.xml` to the standard output.

- Run command to create configuration, storing in the `/etc/gerbera` directory.

```
$ gerbera --create-config | sudo tee /etc/gerbera/config.xml
```

You can start Gerbera with similar command as below:

```
$ gerbera -c /etc/gerbera/config.xml
```

NOTE

- **You might need to create the directory gerbera inside the *~/.config/* folder and change the owner to gerbera**
  - *mkdir ~/.config/gerbera*
  - *sudo chown gerbera:gerbera gerbera*
- Gerbera sets the `<home>` to the runtime user's home by default. Be sure to update accordingly.
- Ensure the **gerbera** user has proper permissions to the `config.xml` file.

Configure Server

These settings define the server configuration, this includes UPnP behavior, selection of database, accounts for the UI as well as installation locations of shared data.

server

```
<server> ... </server>
```

- Required

This section defines the server configuration parameters.

port

```
<port>0</port>
```

- Optional
- Default: **0** *(automatic)*

Specifies the port where the server will be listening for HTTP requests. Note, that because of the implementation in the UPnP SDK only ports above 49152 are supported. The value of zero means, that a port will be automatically selected by the SDK.

ip

```
<ip>192.168.0.23</ip>
```

- Optional
- Default: **ip of the first available interface.**

Specifies the IP address to bind to, by default one of the available interfaces will be selected.

interface

```
<interface>eth0</interface>
```

- Optional
- Default: **first available interface.**

Specifies the interface to bind to, by default one of the available interfaces will be selected.

name

```
<name>Gerbera</name>
```

- Optional
- Default: **Gerbera**

Server friendly name, you will see this on your devices that you use to access the server.

manufacturerURL

```
<manufacturerURL>http://gerbera.io/</manufacturerURL>
```

- Optional
- Default: **http://gerbera.io/**

This tag sets the manufacturer URL of a UPnP device, a custom setting may be necessary to trick some renderers in order to enable special features that otherwise are only active with the vendor implemented server.

modelName

```
<modelName>Gerbera</modelName>
```

- Optional
- Default: **Gerbera**

This tag sets the model name of a UPnP device, a custom setting may be necessary to trick some renderers in order to enable special features that otherwise are only active with the vendor implemented server.

modelNumber

```
<modelNumber>0.9.0</modelNumber>
```

- Optional
- Default: **Gerbera version**

This tag sets the model number of a UPnP device, a custom setting may be necessary to trick some renderers in order to enable special features that otherwise are only active with the vendor implemented server.

serialNumber

```
<serialNumber>1</serialNumber>
```

- Optional
- Default: **1**

This tag sets the serial number of a UPnP device.

presentationURL

```
<presentationURL append-to="ip">80/index.html</presentationURL>
```

- Optional
- Default: **"/"**

The presentation URL defines the location of the servers user interface, usually you do not need to change this however, vendors who want to ship our server along with their NAS devices may want to point to the main configuration page of the device.

attributes:

```
append-to=...
```

- Optional
- Default: **"none"**

The append-to attribute defines how the text in the presentationURL tag should be treated. The allowed values are:

```
append-to="none"
```

Use the string exactly as it appears in the presentationURL tag.

```
append-to="ip"
```

Append the string specified in the presentationURL tag to the ip address of the server, this is useful in a dynamic ip environment where you do not know the ip but want to point the URL to the port of your web server.

```
append-to="port"
```

Append the string specified in the presentationURL tag to the server ip and port, this may be useful if you want to serve some static pages using the built in web server.

udn

```
<udn>uuid:[generated-uuid]</udn>
```

- Required
- Default: **none**

Unique Device Name, according to the UPnP spec it must be consistent throughout reboots. You can fill in something yourself. Review the *Generating Configuration* section of the documentation to see how to use `gerbera` to create a default configuration file.

home

```
<home>/home/your_user_name/.config/gerbera</home>
```

- Required
- Default: **'~/.config/gerbera'**

Server home - the server will search for the data that it needs relative to this directory - basically for the sqlite database file. The gerbera.html bookmark file will also be generated in that directory.

webroot

```
<webroot>/usr/share/gerbera/web</webroot>
```

- Required
- Default: **depends on the installation prefix that is passed to the configure script.**

Root directory for the web server, this is the location where device description documents, UI html and js files, icons, etc. are stored.

serverdir

```
<servedir>/home/myuser/mystuff</servedir>
```

- Optional
- Default: **empty (disabled)**

Files from this directory will be served as from a regular web server. They do not need to be added to the database, but they are also not served via UPnP browse requests. Directory listing is not supported, you have to specify full paths.

Example: The file something.jar is located in `/home/myuser/mystuff/javasubdir/something.jar` on your filesystem. Your ip address is 192.168.0.23, the server is running on port 50500. Assuming the above configuration you could download it by entering this link in your web browser: `http://192.168.0.23:50500/content/serve/javasubdir/something.jar`

alive

```
<alive>180</alive>
```

- Optional
- Default: **180**, *this is according to the UPnP specification.*

Interval for broadcasting SSDP:alive messages

Note: If you experience disconnection problems from your device, e.g. Playstation 4, when streaming videos after about 5 minutes, you can try changing the alive value to 86400 (which is 24 hours)

pc-directory

```
<pc-directory upnp-hide="no"/>
```

- Optional
- Default: **no**

Enabling this option will make the PC-Directory container invisible for UPnP devices.

Note: independent of the above setting the container will be always visible in the web UI!

tmpdir

```
<tmpdir>/tmp/</tmpdir>
```

- Optional
- Default: **/tmp/**

Selects the temporary directory that will be used by the server.

bookmark

```
<bookmark>gerbera.html</bookmark>
```

- Optional
- Default: **gerbera.html**

The bookmark file offers an easy way to access the user interface, it is especially helpful when the server is not configured to run on a fixed port. Each time the server is started, the bookmark file will be filled in with a redirect to the servers current IP address and port. To use it, simply bookmark this file in your browser, the default location is `~/.config/gerbera/gerbera.html`

upnp-string-limit

```
<upnp-string-limit>
```

- Optional

- Default: **disabled**

This will limit title and description length of containers and items in UPnP browse replies, this feature was added as a workaround for the TG100 bug which can only handle titles no longer than 100 characters. A negative value will disable this feature, the minimum allowed value is “4” because three dots will be appended to the string if it has been cut off to indicate that limiting took place.

ui

```
<ui enabled="yes" poll-interval="2" poll-when-idle="no"/>
```

- Optional

This section defines various user interface settings.

WARNING!

The server has an integrated filesystem browser, that means that anyone who has access to the UI can browse your filesystem (with user permissions under which the server is running) and also download your data! If you want maximum security - disable the UI completely! Account authentication offers simple protection that might hold back your kids, but it is not secure enough for use in an untrusted environment!

Note: since the server is meant to be used in a home LAN environment the UI is enabled by default and accounts are deactivated, thus allowing anyone on your network to connect to the user interface.

Attributes:

```
enabled=...
```

- Optional
- Default: **yes**

Enables (”yes”) or disables (”no”) the web user interface.

```
show-tooltips=...
```

- Optional
- Default: **yes**

This setting specifies if icon tooltips should be shown in the web UI.

```
poll-interval=...
```

- Optional
- Default: **2**

The poll-interval is an integer value which specifies how often the UI will poll for tasks. The interval is specified in seconds, only values greater than zero are allowed.

```
poll-when-idle=...
```

- Optional
- Default: **no**

The poll-when-idle attribute influences the behavior of displaying current tasks: - when the user does something in the UI (i.e. clicks around) we always poll for the current task and will display it - if a task is active, we will continue polling in the background and update the current task view accordingly

- when there is no active task (i.e. the server is currently idle) we will stop the background polling and only request updates upon user actions, but not when the user is idle (i.e. does not click around in the UI)

Setting poll-when-idle to "yes" will do background polling even when there are no current tasks; this may be useful if you defined multiple users and want to see the tasks the other user is queuing on the server while you are actually idle.

The tasks that are monitored are:

- adding files or directories
- removing items or containers
- automatic rescans

Child tags:

```
<accounts enabled="yes" session-timeout="30"/>
```

- Optional

This section holds various account settings.

Attributes:

```
enabled=...
```

- Optional
- Default: **yes**

Specifies if accounts are enabled `yes` or disabled `no`.

```
session-timeout=...
```

- Optional
- Default: **30**

The session-timeout attribute specifies the timeout interval in minutes. The server checks every five minutes for sessions that have timed out, therefore in the worst case the session times out after session-timeout + 5 minutes.

Accounts can be defined as shown below:

```
<account user="name" password="password"/>
<account user="name" password="password"/>
```

- Optional

There can be multiple users, however this is mainly a feature for the future. Right now there are no per-user permissions.

```
<items-per-page default="25">
```

- Optional
- Default: **25**

This sets the default number of items per page that will be shown when browsing the database in the web UI. The values for the items per page drop down menu can be defined in the following manner:

```
<option>10</option>
<option>25</option>
<option>50</option>
<option>100</option>
```

Default: 10, 25, 50, 100

Note: this list must contain the default value, i.e. if you define a default value of 25, then one of the `<option>` tags must also list this value.

storage

```
<storage caching="yes">
```

- Required

Defines the storage section - database selection is done here. Currently sqlite3 and mysql are supported. Each storage driver has it's own configuration parameters.

Child Tags

```
caching="yes"
```

- Optional
- Default: **yes**

Enables caching, this feature should improve the overall import speed.

```
<sqlite enabled="yes>
```

- Required **if MySQL is not defined**

Allowed values are `sqlite3` or `mysql`, the available options depend on the selected driver.

```
enabled="yes"
```

- Optional
- Default: **yes**

Below are the sqlite driver options:

```
<database-file>gerbera.db</database-file>
```

- Optional
- Default: **gerbera.db**

The database location is relative to the server's home, if the sqlite database does not exist it will be created automatically.

```
<synchronous>off</synchronous>
```

- Optional
- Default: **off**

Possible values are `off`, `normal` and `full`.

This option sets the SQLite pragma **synchronous**. This setting will affect the performance of the database write operations. For more information about this option see the SQLite documentation: http://www.sqlite.org/pragma.html#pragma_synchronous

```
<on-error>restore</on-error>
```

- Optional
- Default: **restore**

Possible values are `restore` and `fail`.

This option tells Gerbera what to do if an SQLite error occurs (no database or a corrupt database). If it is set to **restore** it will try to restore the database from a backup file (if one exists) or try to recreate a new database from scratch.

If the option is set to **fail**, Gerbera will abort on an SQLite error.

```
<backup enabled="no" interval="6000"/>
```

- Optional

Backup parameters:

```
enabled=...
```

- Optional
- Default: **no**

Enables or disables database backup.

```
interval=...
```

- Optional
- Default: **600**

Defines the backup interval in seconds.

```
<mysql enabled="no"/>
```

Defines the MySQL storage driver section.

```
enabled=...
```

- Optional
- Default: **yes**

Enables or disables the MySQL driver.

Below are the child tags for MySQL:

```
<host>localhost</host>
```

- Optional
- Default: **"localhost"**

This specifies the host where your MySQL database is running.

```
<port>0</port>
```

- Optional
- Default: **0**

This specifies the port where your MySQL database is running.

```
<username>root</username>
```

- Optional
- Default: **"gerbera"**

This option sets the user name that will be used to connect to the database.

```
<password></password>
```

- Optional
- Default: **no password**

Defines the password for the MySQL user. If the tag doesn't exist Gerbera will use no password, if the tag exists, but is empty Gerbera will use an empty password. MySQL has a distinction between no password and an empty password.

```
<database>gerbera</database>
```

- Optional
- Default: **"gerbera"**

Name of the database that will be used by Gerbera.

Configure Extended Runtime Options

extended-runtime-options

```
<extended-runtime-options>
```

These options reside under the server tag and allow to additinally control the so called "runtime options". The difference to the import options is:

Import options are only triggered when data is being imported into the database, this means that if you want new settings to be applied to already imported data, you will have to reimport it. Runtime options can be switched on and off without the need to reimport any media. Unfortunately you still need to restart the server so that these options take effect, however after that they are immediately active.

ffmpegthumbnailer

```
<ffmpegthumbnailer enabled="no">
```

- Optional
- Default: **no**

Ffmpegthumbnailer is a nice, easy to use library that allows to generate thumbnails from video files. Some DLNA compliant devices support video thumbnails, if you think that your device may be one of those you can try enabling this option.

The following options allow to control the ffmpegthumbnailer library (these are basically the same options as the ones offered by the ffmpegthumbnailer command line application). All tags below are optional and have sane default values.

```
<cache-dir enabled="yes">/home/gerbera/cache-dir</cache-dir>
```

- Optional
- Default: **<gerbera-home>/cache-dir**

Storage location for the thumbnail cache when FFMPEGThumbnailer is enabled. Defaults to Gerbera Home. Creates a thumbnail with file format as: `<movie-filename>-thumb.jpg`

The attributes of the tag have the following meaning:

```
enabled=...
```

- Optional
- Default: **yes**

Enables or disables the use of cache directory for thumbnails, set to `yes` to enable the feature.

```
<thumbnail-size>128</thumbnail-size>
```

- Optional
- Default: **128**

The thumbnail size should not exceed 160x160 pixels, higher values can be used but will mostprobably not be supported by DLNA devices. The value of zero or less is not allowed.

```
<seek-percentage>5</seek-percentage>
```

- Optional
- Default: **5**

Time to seek to in the movie (percentage), values less than zero are not allowed.

```
<filmstrip-overlay>yes</filmstrip-overlay>
```

- Optional
- Default: **yes**

Creates a filmstrip like border around the image, turn this option off if you want pure images.

```
<image-quality>8</image-quality>
```

- Optional
- Default: **8**

Sets the image quality of the generated thumbnails.

```
<workaround-bugs>no</workaround-bugs>
```

- Optional

- Default: **no**

According to ffmpegthumbnailer documentation, this option will enable workarounds for bugs in older ffmpeg versions. You can try enabling it if you experience unexpected behaviour, like hangups during thumbnail generation, crashes and alike.

lastfm

```
<lastfm enabled="no">
```

- Optional

Support for the last.fm service.

```
<username>login</username>
```

- Required

Your last.fm user name.

```
<password>pass</password>
```

- Required

Your last.fm password.

```
<mark-played-items enabled="no" suppress-cds-updates="yes">
```

- Optional

The attributes of the tag have the following meaning:

```
enabled=...
```

- Optional
- Default: **no**

Enables or disables the marking of played items, set to `yes` to enable the feature.

```
suppress-cds-updates=...
```

- Optional
- Default: **yes**

This is an advanced feature, leave the default setting if unsure. Usually, when items are modified the system sends out container updates as specified in the Content Directory Service. This notifies the player that data in a particular container has changed, players that support CDS updates will rebrowse the container and refresh the view. However, in this case we probably do not want it (this actually depends on the particular player implementation). For example, if the system updates the list of currently playing items, the player could interrupt playback and rebrowse the current container - clearly an unwanted behaviour. Because of this, Gerbera provides an option to suppress and not send out container updates - only for the case where the item is marked as "played". In order to see the changes you will have to get out of the current container and enter it again - then the view on your player should get updated.

Note: some players (i.e. PS3) cache a lot of data and do not react to container updates, for those players it may be necessary to leave the server view or restart the player in order to update content (same as when adding new data).

The following tag defines how played items should be marked:

```
<string mode="prepend">* </string>
```

- Optional
- Default: \

Specifies what string should be appended or prepended to the title of the object that will be marked as "played".

```
mode=...
```

- Optional
- Default: **prepend**

Specifies how a string should be added to the object's title, allowed values are "append" and "prepend".

```
<mark>
```

- Optional

This subsection allows to list which type of content should get marked. It could also be used with audio and image content, but otherwise it's probably useless. Thefore Gerbera specifies only three supported types that can get marked:

```
<content>audio</content>
<content>video</content>
<content>image</content>
```

You can specify any combination of the above tags to mark the items you want.

Configure Import

The import settings define various options on how to aggregate the content.

import

```
<import hidden-files="no">
```

- Optional

This tag defines the import section.

Attributes:

```
hidden-files="yes|no"
```

- Optional
- Default: **no**

This attribute defines if files starting with a dot will be imported into the database (”yes”). Autoscan can override this attribute on a per directory basis.

Child tags:

filesystem-charset

```
<filesystem-charset>UTF-8</filesystem-charset>
```

- Optional
- Default: **if ‘‘nl_langinfo()‘‘ function is present, this setting will be auto detected based on your system locale, else set to UTF-8**

Defines the charset of the filesystem. For example, if you have file names in Cyrillic KOI8-R encoding, then you should specify that here. The server uses UTF-8 internally, this import parameter will help you to correctly import your data.

metadata-charset

```
<metadata-charset>UTF-8</metadata-charset>
```

- Optional
- Default: **if ‘‘nl_langinfo()‘‘ function is present, this setting will be auto detected based on your system locale, else set to UTF-8**

Same as above, but defines the charset of the metadata (i.e. id3 tags, Exif information, etc.)

scripting script-charset

```
<scripting script-charset="UTF-8">
```

- Optional

Defines the scripting section.

```
script-charset=...
```

- Optional
- Default: **UTF-8**

Below are the available scripting options:

virtual-layout

```
<virtual-layout type="builtin">
```

- Optional

Defines options for the virtual container layout; the so called ”virtual container layout” is the way how the server organizes the media according to the extracted metadata. For example, it allows sorting audio files by Album, Artist, Year and so on.

```
type="builtin|js|disabled"
```

- Optional
- Default: **builtin**

Specifies what will be used to create the virtual layout, possible values are:

- **builtin**: a default layout will be created by the server
- **js**: a user customizable javascript will be used (Gerbera must be compiled with js support)
- **disabled**: only PC-Directory structure will be created, i.e. no virtual layout

The virtual layout can be adjusted using an import script which is defined as follows:

```
<import-script>/path/to/my/import-script.js</import-script>
```

- Required: **if virtual layout type is "js"**
- Default: `${prefix}/share/gerbera/js/import.js`, **where ${prefix} is your installation prefix directory.**

Points to the script invoked upon media import. For more details read about *scripting*

common-script

```
<common-script>/path/to/my/common-script.js</common-script>
```

- Optional
- Default: `${prefix}/share/gerbera/js/common.js`, **where ${prefix} is your installation prefix directory.**

Points to the so called common script - think of it as a custom library of js helper functions, functions added there can be used in your import and in your playlist scripts. For more details read *scripting*

playlist-script

```
<playlist-script create-link="yes">/path/to/my/playlist-script.js</playlist-script>
```

- Optional
- Default: `${prefix}/share/gerbera/js/playlists.js`, **where ${prefix} is your installation prefix directory.**

Points to the script that is parsing various playlists, by default parsing of pls and m3u playlists is implemented, however the script can be adapted to parse almost any kind of text based playlist. For more details read *scripting*

```
create-link="yes|no"
```

- Optional
- Default: **yes**

Links the playlist to the virtual container which contains the expanded playlist items. This means, that if the actual playlist file is removed from the database, the virtual container corresponding to the playlist will also be removed.

magic-file

```
<magic-file>/path/to/my/magic-file</magic-file>
```

- Optional
- Default: **System default**

Specifies an alternative file for filemagic, containing mime type information.

autoscan

```
<autoscan use-inotify="auto">
```

- Optional

Specifies a list of default autoscan directories.

This section defines persistent autoscan directories. It is also possible to define autoscan directories in the UI, the difference is that autoscan directories that are defined via the config file can not be removed in the UI. Even if the directory gets removed on disk, the server will try to monitor the specified location and will re add the removed directory if it becomes available/gets created again.

```
use-inotify="yes|no|auto"
```

- Optional
- Default: **auto**

Specifies if the inotify autoscan feature should be enabled. The default value is `auto`, which means that availability of inotify support on the system will be detected automatically, it will then be used if available. Setting the option to 'no' will disable inotify even if it is available. Allowed values: "yes", "no", "auto"

Child tags:

```
<directory location="/media" mode="timed" interval="3600"
  recursive="no" hidden-files="no"/>
<directory location="/audio" mode="inotify"
  recursive="yes" hidden-files="no"/>
```

- Optional

Defines an autoscan directory and it's parameters.

The attributes specify various autoscan options:

```
location=...
```

- Required

Absolute path to the directory that shall be monitored.

```
mode="inotify|timed"
```

- Required

Scan mode, currently `inotify` and `timed` are supported. Timed mode rescans the given directory in specified intervals, inotify mode uses the kernel inotify mechanism to watch for filesystem events.

```
interval=...
```

- Required: **for "timed" mode**

Scan interval in seconds.

```
recursive="yes|no"
```

- Required

Values of `yes` or `no` are allowed, specifies if autoscan shall monitor the given directory including all sub directories.

```
hidden-files="yes|no"
```

- Optional
- Default: **value specified in <import hidden-files=""/>**

Allowed values: `yes` or `no`, process hidden files, overrides the hidden-files value in the `<import/>` tag.

mappings

```
<mappings>
```

- Optional

Defines various mapping options for importing media, currently two subsections are supported.

This section defines mime type and upnp:class mappings, it is vital if filemagic is not available - in this case media type auto detection will fail and you will have to set the mime types manually by matching the file extension. It is also helpful if you want to override auto detected mime types or simply skip filemagic processing for known file types.

extension-mimetype

```
<extension-mimetype ignore-unknown="no" case-sensitive="no">
```

- Optional

This section holds the file name extension to mime type mappings.

Attributes:

```
ignore-unknown=...
```

- Optional
- Default: **no**

If ignore-unknown is set to "yes", then only the extensions that are listed in this section are imported.

```
case-sensitive=...
```

- Optional
- Default: **no**

Specifies if extensions listed in this section are case sensitive, allowed values are “yes” or “no”.

Child tags:

map

```
<map from="mp3" to="audio/mpeg"/>
```

- Optional

Specifies a mapping from a certain file name extension (everything after the last dot “.”) to mime type.

Note: this improves the import speed, because invoking libmagic to discover the right mime type of a file is omitted for files with extensions listed here.

Note: extension is case sensitive, this will probably need to be fixed.

mime-type-upnpclass

```
<mimetype-upnpclass>
```

- Optional

This section holds the mime type to upnp:class mappings.

Child tags:

map

```
<map from="audio/*" to="object.item.audioItem.musicTrack"/>
```

- Optional

Specifies a mapping from a certain mime type to upnp:class in the Content Directory. The mime type can either be entered explicitly “audio/mpeg” or using a wildcard after the slash `audio/\*`. The values of **from** and **to** attributes are case sensitive.

mimetype-contenttype

```
<mimetype-contenttype>
```

- Optional

This section makes sure that the server knows about remapped mimetypes and still extracts the metadata correctly. For example, we know that id3lib can only handle mp3 files, the default mimetype of mp3 content is audio/mpeg. If the user remaps mp3 files to a different mimetype, we must know about it so we can still pass this item to id3lib for metadata extraction.

Note: if this section is not present in your config file, the defaults will be filled in automatically. However, if you add an empty tag, without defining the following `<treat>` tags, the server assumes that you want to have an empty list and no files will be process by the metadata handler.

treat

```
<treat mimetype="audio/mpeg" as="mp3"/>
```

- Optional

Tells the server what content the specified mimetype actually is.

Note: it makes no sense to define 'as' values that are not below, the server only needs to know the content type of the ones specified, otherwise it does not matter.

The `as` attribute can have following values:

Mapping Table

mimetype	as	Note
audio/mpeg	mp3	The content is an mp3 file and should be processed by either id3lib or taglib (if available).
application/ogg	ogg	The content is an ogg file and should be processed by taglib (if available).
audio/x-flac	flac	The content is a flac file and should be processed by taglib (if available).
image/jpeg	jpg	The content is a jpeg image and should be processed by libexif (if available).
audio/x-mpegurl or audio/x-scpls	playlist	The content is a playlist and should be processed by the playlist parser script.
audio/L16 or audio/x-wav	pcm	The content is a PCM file.
video/x-msvideo	avi	The content is an AVI container, FourCC extraction will be attempted.

library-options

```
<library-options>
```

- Optional

This section holds options for the various supported import libraries, it is useful in conjunction with virtual container

scripting, but also allows to tune some other features as well.

Currently the **library-options** allow additional extraction of the so called auxilary data (explained below) and provide control over the video thumbnail generation.

Here is some information on the auxdata: UPnP defines certain tags to pass along metadata of the media (like title, artist, year, etc.), however some media provides more metadata and exceeds the scope of UPnP. This additional metadata can be used to fine tune the server layout, it allows the user to create a more complex container structure using a customized import script. The metadata that can be extracted depends on the library, currently we support libebexif which provides a default set of keys that can be passed in the options below. The data according to those keys will the be extracted from the media and imported into the database along with the item. When processing the item, the import script will have full access to the gathered metadata, thus allowing the user to organize the data with the use of the extracted information. A practical example would be: if have more than one digital camera in your family you could extract the camera model from the Exif tags and sort your photos in a structure of your choice, like:

- Photos/MyCamera1/All Photos
- Photos/MyCamera1/Date
- Photos/MyCamera2/All Photos
- Photos/MyCamera2/Date

etc.

Child tags:

libexif

```
<libexif>
```

- Optional

Options for the exif library.

Child tags:

auxdata

```
<auxdata>
```

- Optional

Currently only adding keywords to auxdata is supported. For a list of keywords/tags see the libexif documentation. Auxdata can be read by the import java script to gain more control over the media structure.

Child tags:

add-data

```
<add-data tag="keyword1"/>
<add-data tag="keyword2"/>
...
```

- Optional

If the library was able to extract the data according to the given keyword, it will be added to auxdata. You can then use that data in your import scripts.

A sample configuration for the example described above would be:

```
<libexif>
    <auxdata>
        <add-data tag="EXIF_TAG_MODEL"/>
    </auxdata>
</libexif>
```

id3

```
<id3>
```

- Optional

These options apply to id3lib or taglib libraries.

Child tags:

auxdata

```
<auxdata>
```

- Optional

Currently only adding keywords to auxdata is supported. The keywords are those defined in the id3 specification, we do not perform any extra checking, so you could try to use any string as a keyword - if it does not exist in the tag nothing bad will happen.

Here is a list of some possible keywords:

- ID3v2 / MP3

TALB, TBPM, TCOM, TCON, TCOP, TDAT, TDLY, TENC, TEXT, TFLT, TIME, TIT1, TIT2, TIT3, TKEY, TLAN, TLEN, TMED, TOAL, TOFN, TOLY, TOPE, TORY, TOWN, TPE1, TPE2, TPE3, TPE4, TPOS, TPUB, TRCK, TRDA, TRSN, TRSO, TSIZ, TSRC, TSSE, TYER, TXXX:CATALOGNUMBER, TXXX:MusicBrainz Album Type, ...

- Vorbis / FLAC

ALBUMSORT, COMPOSER, ENCODEDBY, MUSICBRAINZ_ARTISTID, CATALOGNUMBER, RELEASETYPE, ...

- any other user defined keyword, e.g. for APEv2 or iTunes MP4

  Child tags:

add-data

```
<add-data tag="TCOM"/>
<add-data tag="COMPOSER"/>
<add-data tag="TENC"/>
<add-data tag="ENCODEDBY"/>
...
```

- Optional

If the library was able to extract the data according to the given keyword, it will be added to auxdata. You can then use that data in your import scripts.

A sample configuration for the example described above would be:

```
<id3>
    <auxdata>
        <add-data tag="TCOM"/>
        <add-data tag="COMPOSER"/>
        <add-data tag="TENC"/>
        <add-data tag="ENCODEDBY"/>
    </auxdata>
</id3>
```

Configure Online Content

This section resides under import and defines options for various supported online services.

Example of online content XML configuration

```
<import>
    <online-content>
        <AppleTrailers enabled="[yes,no]" refresh="[seconds]"
            purge-after="[seconds]"
            update-at-start="[yes,no]"
            resolution="[640,720]"/>
        <SopCast enabled="[yes,no]" refresh="[seconds]"
            purge-after="[seconds]"
            update-at-start="[yes,no]"/>
    </online-content>
</import>
```

online-content

```
<online-content fetch-buffer-size="262144" fetch-buffer-fill-size="0">
```

- Optional

This tag defines the online content section.

Attributes:

```
fetch-buffer-size=...
```

- Optional
- Default: **262144**

Often, online content can be directly accessed by the player - we will just give it the URL. However, sometimes it may be necessary to proxy the content through Gerbera. This setting defines the buffer size in bytes, that will be used when fetching content from the web. The value must not be less than allowed by the curl library (usually 16384 bytes).

```
fetch-buffer-fill-size=...
```

- Optional
- Default: **0 (disabled)**

This setting allows to prebuffer a certain amount of data, given in bytes, before sending it to the player, this should ensure a constant data flow in case of slow connections. Usually this setting is not needed, because most players will anyway have some kind of buffering, however if the connection is particularly slow you may want to try enable this setting.

AppleTrailers

```
<AppleTrailers enabled="[yes,no]"
    refresh="[seconds]" update-at-start="[yes,no]" resolution="[640,720]"/>
```

- Optional

This tag defines the online content for Apple Trailers

Attributes

```
enabled=...
```

- Default: **no**

```
refresh=...
```

- Default: **43200**

The amount of time to wait before refreshing the online content

```
update-at-start=...
```

- Default: **no**

Upon starting Gerbera, the Apple Trailers content will be refreshed.

```
resolution=...
```

- Default: **720**

Sets the Apple Trailers URL to retrieve the content, affecting the resolution size that is downloaded.

SopCast

```
<SopCast enabled="[yes,no]" refresh="<int>" purge-after="<int>" update-at-start="[yes,
↪no]"/>
```

- Optional

This tag defines the online content for SopCast

Attributes

```
enabled=...
```

- Default: **no**

```
refresh=...
```

- Default: **43200**

The amount of time to wait before refreshing the online content

```
update-at-start=...
```

- Default: **no**

Upon starting Gerbera, the SopCast content will be refreshed.

```
purge-after=...
```

- Default: **0**

Sets the expiration time of downloaded content in seconds.

Configure Transcoding

The transcoding section allows to define ways on how to transcode content.

transcoding

```
<transcoding enabled="yes" fetch-buffer-size="262144" fetch-buffer-fill-size="0">
```

- Optional

This tag defines the transcoding section.

Attributes:

```
enabled=...
```

- Optional
- Default: **yes**

This attribute defines if transcoding is enabled as a whole, possible values are ”yes” or ”no”.

```
fetch-buffer-size=...
```

- Optional
- Default: **262144**

In case you have transcoders that can not handle online content directly (see the accept-url parameter below), it is possible to put the transcoder between two FIFOs, in this case Gerbera will fetch the online content. This setting defines the buffer size in bytes, that will be used when fetching content from the web. The value must not be less than allowed by the curl library (usually 16384 bytes).

```
fetch-buffer-fill-size=...
```

- Optional
- Default: **0 (disabled)**

This setting allows to prebuffer a certain amount of data before sending it to the transcoder, this should ensure a constant data flow in case of slow connections. Usually this setting is not needed, because most transcoders will just patiently wait for data and we anyway buffer on the output end. However, we observed that ffmpeg will fail to transcode flv files if it encounters buffer underruns - this setting helps to avoid this situation.

Child tags:

mimetype-profile-mappings

```
<mimetype-profile-mappings>
```

The mime type to profile mappings define which mime type is handled by which profile.

Different mime types can map to the same profile in case that the transcoder in use supports various input formats. The same mime type can also map to several profiles, in this case multiple resources in the XML will be generated, allowing the player to decide which one to take.

The mappings under mimetype-profile are defined in the following manner:

transcode

```
<transcode mimetype="audio/x-flac" using="oggflac-pcm"/>
```

- Optional

In this example we want to transcode our flac audio files (they have the mimetype audio/x-flac) using the "oggflac-pcm" profile which is defined below.

```
mimetype=...
```

Selects the mime type of the source media that should be transcoded.

```
using=...
```

Selects the transcoding profile that will handle the mime type above. Information on how to define transcoding profiles can be found below.

profiles

```
<profiles>
```

This section defines the various transcoding profiles.

```
<profile name="oggflag-pcm" enabled="yes" type="external">
```

- Optional

Definition of a transcoding profile.

```
name=...
```

- Required

Name of the transcoding profile, this is the name that is specified in the mime type to profile mappings.

```
enabled=...
```

- Required

Enables or disables the profile, allowed values are ”yes” or ”no”.

```
type=...
```

- Required

Defines the profile type, currently only ”external” is supported, this will change in the future.

```
<mimetype>audio/x-wav</mimetype>
```

- Required

Defines the mime type of the transcoding result (i.e. of the transcoded stream). In the above example we transcode to PCM.

```
<accept-url>yes</accept-url>
```

- Optional
- Default: **yes**

Some transcoders are able to handle non local content, i.e. instead giving a local file name you can pass an URL to the transcoder. However, some transcoders can only deal with local files, for this case set the value to ”no”.

```
<first-resource>no</first-resource>
```

- Optional
- Default: **no**

It is possible to offer more than one resource in the browse result, a good player implementation will go through all resources and pick the one that it can handle best. Unfortunately most players only look at the first resource and ignore the rest. When you add a transcoding profile for a particular media type it will show up as an additional resource in the browse result, using this parameter you can make sure that the transcoded resource appears first in the list.

Note: if more than one transcoding profile is applied on one source media type (i.e. you transcode an OGG file to MP3 and to PCM), and the first-resource parameter is specified in both profiles, then the resource positions are undefined.

```
<hide-original-resource>no</hide-original-resource>
```

- Optional
- Default: **no***

This parameter will hide the resource of the original media when sending the browse result to the player, this can be useful if your device gets confused by multiple resources and allows you to send only the transcoded one.

```
<accept-ogg-theora>no</accept-org-theora>
```

- Optional

- Default: **no***

As you may know, OGG is just a container, the content could be Vorbis or Theora while the mime type is "application/ogg". For transcoding we need to identify if we are dealing with audio or video content, specifying yes in this tag in the profile will make sure that only OGG files containing Theora will be processed.

```
<avi-fourcc-list mode="ignore">
```

- Optional
- Default: **disabled***

This option allows to specify a particular list of AVI fourcc strings that can be either set to be ignored or processed by the profile.

Note: this option has no effect on non AVI content.

```
mode=...
```

- Required

Specifies how the list should be handled by the transcoding engine, possible values are:

```
"disabled"
```

The option is completely disabled, fourcc list is not being processed.

```
"process"
```

Only the fourcc strings that are listed will be processed by the transcoding profile, AVI files with other fourcc strings will be ignored. Setting this is useful if you want to transcode only some specific fourcc's and not transcode the rest.

```
"ignore"
```

The fourcc strings listed will not be transcoded, all other codecs will be transcoded. Setting this might be useful if you want to prevent a limited number of codecs from being transcoded, but want to apply transcoding on the rest (i.e. - do not transcode divx and xvid, but want to transcode mjpg and whatever else might be in the AVI container).

The list of fourcc strings is enclosed in the avi-fourcc-list section:

```
<fourcc>XVID</fourcc>
<fourcc>DX50</fourcc>
```

etc...

```
   <agent command="ogg123" arguments="-d wav -f %out %in"/>

* Required
```

Defines the transcoding agent and the parameters, in the example above we use ogg123 to convert ogg or flac to wav.

```
command=...
```

- Required

Defines the transcoder binary that will be executed by Gerbera upon a transcode request, the binary must be in $PATH. It is very important that the transcoder is capable of writing the output to a FIFO, some applications, for example ffmpeg, have problems with that. The command line arguments are specified separately (see below).

```
arguments=...
```

- Required

Specifies the command line arguments that will be given to the transcoder application upon execution. There are two special tokens:

```
%in
%out
```

Those tokens get substituted by the input file name and the output FIFO name before execution.

```
<buffer size="1048576" chunk-size="131072" fill-size="262144"/>
```

- Required

These settings help you to achieve a smooth playback of transcoded media. The actual values need to be tuned and depend on the speed of your system. The general idea is to buffer the data before sending it out to the player, it is also possible to delay first playback until the buffer is filled to a certain amount. The prefill should give you enough space to overcome some high bitrate scenes in case your system can not transcode them in real time.

```
size=...
```

- Required

Size of the buffer in bytes.

```
chunk-size=...
```

- Required

Size of chunks in bytes, that are read by the buffer from the transcoder. Smaller chunks will produce a more constant buffer fill ratio, however too small chunks may slow things down.

```
fill-size=...
```

- Required

Initial fill size - number of bytes that have to be in the buffer before the first read (i.e. before sending the data to the player for the first time). Set this to 0 (zero) if you want to disable prefilling.

```
<resolution>320x240</resolution>
```

- Optional
- Default: **not specified**

Allows you to tell the resolution of the transcoded media to your player. This may be helpful if you want to generate thumbnails for your photos, or if your player has the ability to pick video streams in a particular resolution. Of course the setting should match the real resolution of the transcoded media.

```
<use-chunked-encoding>yes</use-chunked-encoding>
```

- Optional
- Default: **yes**

Specifies that the content should be sent out using chunked HTTP encoding, this is the default setting for transcoded streams, because the content length of the data is not known.

```
<sample-frequency>source</sample-frequency>
```

- Optional
- Default: **source**

Specifies the sample frequency of the transcoded media, this information is passed to the player and is particularly important when streaming PCM data. Possible values are:

- **source** - automatically set the same frequency as the frequency of the source content, which is useful if you are not doing any resampling
- **off** - do not provide this information to the player
- **frequency** - specify a fixed value, where *frequency* is a numeric value > 0

```
<audio-channels>source</audio-channels>
```

- Optional
- Default: **source**

Specifies the number of audio channels in the transcoded media, this information is passed to the player and is particularly important when streaming PCM data. Possible values are:

- **source** - automatically set the same number of audio channels as in the source content
- **off** - do not provide this information to the player
- **number** - specify a fixed value, where *number* is a numeric value > 0

```
<thumbnail>yes</thumbnail>
```

- Optional
- Default: **no**

Note: this is an experimental option, the implementation will be refined in the future releases.

This is a special option which was added for the PS3 users. If the resolution option (see above) was set, then, depending on the resolution an special DLNA tag will be added, marking the resource as a thumbnail. This is useful if you have a transcoding script that extracts an image out of the video and presents it as a thumbnail.

Use the option with caution, no extra checking is being done if the resulting mimetype represents an image, also, it is will only work if the output of the profile is a JPG image.

4.2.9 Compile Gerbera

Gerbera uses the excellent CMake build system.

Dependencies

Note: Remember to install associated development packages, because development headers are needed for compilation!

In order to compile Gerbera you will have to install the following packages:

Library	Version	Required?	Note	Compile-time option	Default
libupnp	>=1.8.6	Required	pupnp		
libuuid		Depends on OS	On BSD native libuuid is used others require e2fsprogs-libuuid		
pugixml		Required			
fmtlib		Required			
libiconv		Required			
sqlite3		Required	Database storage		
duktape		Optional	Scripting Support	WITH_JS	Enabled
mysql		Optional	Alternate database storage	WITH_MYSQL	Disabled
curl		Optional	Enables web services	WITH_CURL	Enabled
taglib	1.11.1	Optional	Audio tag support	WITH_TAGLIB	Enabled
libmagic		Optional	File type detection	WITH_MAGIC	Enabled
ffmpeg/libav		Optional	File metadata	WITH_AVCODEC	Disabled
libexif		Optional	JPEG Exif metadata	WITH_EXIF	Enabled
libexiv2		Optional	Exif, IPTC, XMP metadata	WITH_EXIV2	Disabled
lastfmlib	0.4.0	Optional	Enables scrobbling	WITH_LASTFM	Disabled
ffmpegthumbnailer		Optional	Generate video thumbnails	WITH_FFMPEGTHUMBNAILER	Disabled
inotify		Optional	Efficient file monitoring	WITH_INOTIFY	Enabled

Quick Start Build

```
git clone https://github.com/gerbera/gerbera.git
mkdir build
cd build
cmake ../gerbera -DWITH_MAGIC=1 -DWITH_MYSQL=1 -DWITH_CURL=1 -DWITH_JS=1 \
-DWITH_TAGLIB=1 -DWITH_AVCODEC=1 -DWITH_FFMPEGTHUMBNAILER=1 -DWITH_EXIF=1 -DWITH_
↪LASTFM=1
make -j4
sudo make install
```

Alternatively, the options can be set using a GUI (make sure to press "c" to configure after toggling settings in the GUI):

```
git clone https://github.com/gerbera/gerbera.git
mkdir build
cd build
cmake ../gerbera
make edit_cache
# Enable some of the WITH... options
make -j4
sudo make install
```

Build On Ubuntu 16.04

```
apt-get install uuid-dev libsqlite3-dev libmysqlclient-dev \
libmagic-dev libexif-dev libcurl4-openssl-dev libspdlog-dev libpugixml-dev
# If building with LibAV/FFmpeg (-DWITH_AVCODEC=1)
apt-get install libavutil-dev libavcodec-dev libavformat-dev libavdevice-dev \
libavfilter-dev libavresample-dev libswscale-dev libswresample-dev libpostproc-dev
```

The following packages are too old in 16.04 and must be installed from source: **taglib (1.11.x)**, and **libupnp (1.8.x).**

libupnp must be configured/built with `--enable-ipv6`. See `scripts/install-pupnp18.sh` for details.

Build On FreeBSD

The following has been tested on FreeBSD 11.0 using a clean jail environment.

1. Install the required *prerequisites* as root using either ports or packages. This can be done via Package manager or ports. (pkg manager is used here.) Include mysql if you wish to use that instead of SQLite3.

```
pkg install wget git autoconf automake libtool taglib cmake gcc libav ffmpeg \
libexif pkgconf liblastfm gmake
```

2. Clone repository, build depdences in current in ports and then build gerbera.

```
git clone https://github.com/gerbera/gerbera.git
mkdir build
cd build
sh ../gerbera/scripts/install-pupnp18.sh
sh ../gerbera/scripts/install-duktape.sh
cmake ../gerbera -DWITH_MAGIC=1 -DWITH_MYSQL=0 -DWITH_CURL=1 -DWITH_JS=1 -DWITH_
↪TAGLIB=1 -DWITH_AVCODEC=1 \
-DWITH_EXIF=1 -DWITH_LASTFM=0 -DWITH_SYSTEMD=0
make -j4
sudo make install
```

Build On macOS

The following has been tested on macOS High Sierra 10.13.4

The Gerbera Team maintains a Homebrew Tap to build and install Gerbera Media Server. Take a look at the Homebrew formula to see an example of how to compile Gerbera on macOS.

homebrew-gerbera/gerbera.rb

4.3 Index

- genindex

Index

P

Q

R

S

T

U

W