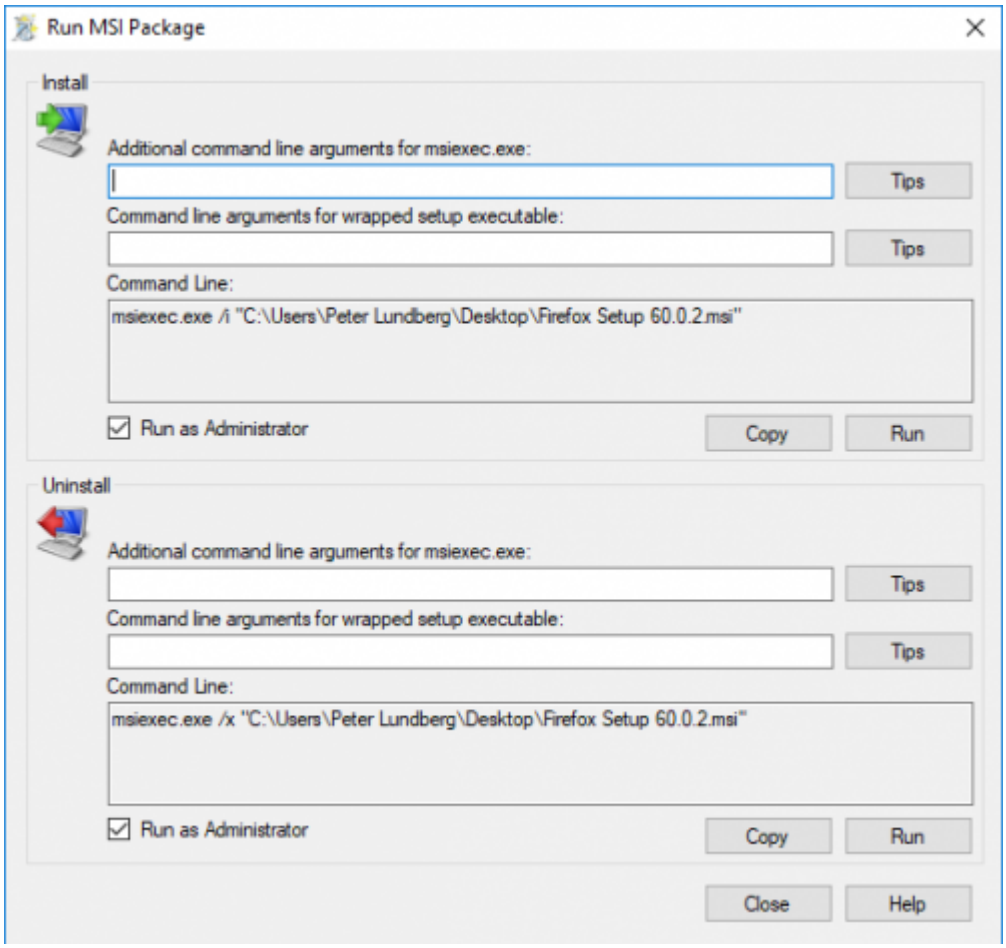
Installing Using Msi

Step-by-step guide to install a "Hello World" application using an MSI (Microsoft Installer) file:

1. Download MSI File: Obtain the MSI file for the "Hello World" application. This file typically has a .msi extension.
2. Locate MSI File: Once downloaded, locate the MSI file on your computer. It's often found in the Downloads folder unless you specified a different location.
3. Double-click the MSI File: Double-click on the MSI file to start the installation process. This action typically launches the Windows Installer, which manages the installation.
4. User Account Control (UAC) Prompt: Depending on your Windows settings, you may see a User Account Control prompt asking for permission to make changes to your device. Click "Yes" to proceed with the installation.
5. Welcome Screen: The installation wizard will open with a welcome screen. Click "Next" to continue.
6. License Agreement: Read the license agreement carefully. If you agree to the terms, select the option to accept the agreement and click "Next."
7. Destination Folder: Choose the destination folder where you want to install the application. The default location is usually fine for most users. Click "Next" to proceed.
8. Installation: Click on the "Install" button to start installing the application. The installer will copy the necessary files to your computer.
9. Progress Bar: You'll see a progress bar indicating the status of the installation. Wait for the process to complete.
10. Completion: Once the installation is finished, you'll see a completion screen. This screen may offer options like launching the application or viewing the readme file. You can choose to launch the application immediately or simply click "Finish" to exit the installer.
11. Verification: After installation, you can verify that the application has been installed correctly by checking your Start menu or desktop shortcuts, if any, or by searching for the application in the Windows search bar.



Windows (x64)

Windows installers and builds in ZIP archive format are available here.

> **WARNING**
>
> If you installed a version prior to 10.4.0 using a PowerShell script, you will need to manually remove the service using the command `nssm remove HelloWorld` and uninstall the server by remove all the files manually. Also one might need to move the data files to the correct location, or point the installer at the old location.

> **WARNING**
>
> The Basic Install is the recommended way to run the HelloWorld Server. Using the Advanced/Service mode may experience FFmpeg hardware acceleration issues, and is only for advanced users.

Install using Installer (x64)

Install

1. Download the latest version.
2. Run the installer.
3. (Optional) When installing as a service (not recommended), pick the service account type.
4. If everything was completed successfully, HelloWorld is now running.
5. Open your browser at http://your_local_IP_address:8096 to finish setting up HelloWorld.

Update

1. Download the latest version.
2. Close or Stop HelloWorld if it is running.
3. Run the installer.
4. If everything was completed successfully, the new version is installed.

Uninstall

1. Go to `Add or remove programs` in Windows.
2. Search for HelloWorld.
3. Click Uninstall.

Manual Installation (x86/x64)

Install

1. Download and extract the latest version.
2. Create a folder `HelloWorld` at your preferred install location.
3. Copy the extracted folder into the `HelloWorld` folder and rename it to `system`.
4. Create `HelloWorld.bat` within your `HelloWorld` folder containing:
    - To use the default library/data location at `%localappdata%`:

```
<--Your install path-->\HelloWorld\system\HelloWorld.exe
```

    - To use a custom library/data location (Path after the -d parameter):

```
<--Your install path-->\HelloWorld\system\HelloWorld.exe -d <--Your install path-
->\HelloWorld\data
```

    - To use a custom library/data location (Path after the -d parameter) and disable the auto-start of the webapp:

```
<--Your install path-->\HelloWorld\system\HelloWorld.exe -d <--Your install path--
>\HelloWorld\data -noautorunwebapp
```

5. Run

```
HelloWorld.bat
```

6. Open your browser at `http://<--Server-IP-->:8096` (if auto-start of webapp is disabled)

Update

1. Stop HelloWorld
2. Rename the HelloWorld `system` folder to `system-bak`
3. Download and extract the latest HelloWorld version
4. Copy the extracted folder into the `HelloWorld` folder and rename it to `system`
5. Run `HelloWorld.bat` to start the server again

Rollback

1. Stop HelloWorld.
2. Delete the `system` folder.
3. Rename `system-bak` to `system`.
4. Run `HelloWorld.bat` to start the server again.

macOS

macOS Application packages and builds in TAR archive format are available here.

Install

1. Download the latest version.
2. Drag the `.app` package into the Applications folder.
3. Start the application.
4. Click the icon in the menu bar and select "Launch Web UI".

Upgrade

1. Download the latest version.
2. Stop the currently running server either via the dashboard or using the menu bar icon.
3. Drag the new `.app` package into the Applications folder and click yes to replace the files.
4. Start the application.

Uninstall

1. Stop the currently running server either via the dashboard or using the application icon.
2. Move the `.app` package to the trash.

Deleting Configuration

This will delete all settings and user information. This applies for the .app package and the portable version.

1. Delete the folder `~/.config/HelloWorld/`
2. Delete the folder `~/.local/share/HelloWorld/`

Portable Version

1. Download the latest version
2. Extract it into the Applications folder
3. Open Terminal and type `cd` followed with a space then drag the HelloWorld folder into the terminal.
4. Type `./HelloWorld` to run HelloWorld.
5. Open your browser at http://localhost:8096

Closing the terminal window will end HelloWorld. Running HelloWorld in screen or tmux can prevent this from happening.

Upgrading the Portable Version

1. Download the latest version.
2. Stop the currently running server either via the dashboard or using `CTRL+C` in the terminal window.
3. Extract the latest version into Applications
4. Open Terminal and type `cd` followed with a space then drag the HelloWorld folder into the terminal.
5. Type `./HelloWorld` to run HelloWorld.
6. Open your browser at http://localhost:8096

Uninstalling the Portable Version

1. Stop the currently running server either via the dashboard or using `CTRL+C` in the terminal window.
2. Move `/Application/HelloWorld-version` folder to the Trash. Replace version with the actual version number you are trying to delete.

Using FFmpeg with the Portable Version

The portable version doesn't come with FFmpeg by default, so to install FFmpeg you have three options.

- use the package manager homebrew by typing `brew install ffmpeg` into your Terminal (here's how to install homebrew if you don't have it already
- download the most recent static build from this link (compiled by a third party see this page for options and information), or
- compile from source available from the official website

More detailed download options, documentation, and signatures can be found.

If using static build, extract it to the `/Applications/` folder.

Navigate to the Playback tab in the Dashboard and set the path to FFmpeg under FFmpeg Path.

Linux

Linux (generic amd64)

Generic amd64, arm64, and armhf Linux builds in TAR archive format are available here.

Base Installation Process

Create a directory in `/opt` for HelloWorld and its files, and enter that directory.

```
sudo mkdir /opt/HelloWorld
cd /opt/HelloWorld
```

Download the latest generic Linux build for your architecture. The rest of these instructions assume version 10.8.1 is being installed (i.e. `HelloWorld_10.8.1_amd64.tar.gz`). Download the generic build, then extract the archive:

```
sudo wget
https://repo.HelloWorld.org/releases/server/linux/stable/combined/HelloWorld_10.8.1_amd64.ta
r.gz
sudo tar xvzf HelloWorld_10.8.1_amd64.tar.gz
```

Create a symbolic link to the HelloWorld 10.8.1 directory. This allows an upgrade by repeating the above steps and enabling it by simply re-creating the symbolic link to the new version.

```
sudo ln -s HelloWorld_10.8.1 HelloWorld
```

Create four sub-directories for HelloWorld data.

```
sudo mkdir data cache config log
```

`ffmpeg` Installation

If you are not running a Debian derivative, install `ffmpeg` through your OS's package manager, and skip this section.

> **WARNING**
>
> Not being able to use `HelloWorld-ffmpeg5` will most likely break hardware acceleration and tonemapping.

If you are running Debian or a derivative, you should download and install an `ffmpeg` release built specifically for HelloWorld. Be sure to download the latest release that matches your OS (`5.0.1-8` for Debian Bullseye assumed below).

```
sudo wget https://repo.HelloWorld.org/releases/server/debian/versions/HelloWorld-ffmpeg/5.0.1-8/HelloWorld-ffmpeg5_5.0.1-8-bullseye_amd64.deb
sudo dpkg --install HelloWorld-ffmpeg5_5.0.1-8-bullseye_amd64.deb
```

If you run into any dependency errors, run this and it will install them and `HelloWorld-ffmpeg5`.

```
sudo apt install -f
```

Running HelloWorld

Due to the number of command line options that must be passed, it is easiest to create a small script to run HelloWorld.

```
sudo nano HelloWorld.sh
```

Then paste the following commands and modify as needed.

```
#!/bin/bash
HelloWorldDIR="/opt/HelloWorld"
FFMPEGDIR="/usr/share/HelloWorld-ffmpeg"

$HelloWorldDIR/HelloWorld/HelloWorld \
 -d $HelloWorldDIR/data \
 -C $HelloWorldDIR/cache \
 -c $HelloWorldDIR/config \
 -l $HelloWorldDIR/log \
 --ffmpeg $FFMPEGDIR/ffmpeg
```

Assuming you desire HelloWorld to run as a non-root user, `chmod` all files and directories to your normal login user and group. Also make the startup script above executable.

```
sudo chown -R user:group *
sudo chmod u+x HelloWorld.sh
```

Finally you can run it. You will see lots of log information when run, this is normal. Setup is as usual in the web browser.

```
./HelloWorld.sh
```

Starting HelloWorld on boot (optional)

Create a `systemd` unit file.

```
cd /etc/systemd/system
sudo nano HelloWorld.service
```

Then paste the following contents, replacing `youruser` with your username.

```
[Unit]
Description=HelloWorld
After=network.target

[Service]
Type=simple
User=youruser
Restart=always
ExecStart=/opt/HelloWorld/HelloWorld.sh

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

Apply the correct permissions to the file, enable the service to start on boot, then start it.

```
sudo chmod 644 HelloWorld.service
sudo systemctl daemon-reload
sudo systemctl enable HelloWorld.service
sudo systemctl start HelloWorld.service
```

Portable DLL

Platform-agnostic .NET Core DLL builds in TAR archive format are available here. These builds use the binary `HelloWorld.dll` and must be loaded with `dotnet`.

Arch Linux

HelloWorld can be found in the AUR as HelloWorld, HelloWorld-bin and HelloWorld-git.

Fedora

Fedora builds in RPM package format are available here for now but an official Fedora repository is coming soon.

1. You will need to enable rpmfusion as ffmpeg is a dependency of the HelloWorld server package

```
sudo dnf install https://mirrors.rpmfusion.org/free/fedora/rpmfusion-free-release-$(rpm
-E %fedora).noarch.rpm https://mirrors.rpmfusion.org/nonfree/fedora/rpmfusion-nonfree-
release-$(rpm -E %fedora).noarch.rpm
```

> **NOTE**
> You do not need to manually install ffmpeg, it will be installed by the HelloWorld server package as a dependency

2. Install the HelloWorld server

```
sudo dnf install (link to version HelloWorld server you want to install)
```

3. Install the HelloWorld web interface

```
sudo dnf install (link to web RPM you want to install)
```

4. Enable HelloWorld service with systemd

```
sudo systemctl start HelloWorld
```

```
sudo systemctl enable HelloWorld
```

5. Open HelloWorld service with firewalld

```
sudo firewall-cmd --permanent --add-service=HelloWorld
```

> **NOTE**
>
> This will open the following ports 8096 TCP used by default for HTTP traffic, you can change this in the dashboard 8920 TCP used by default for HTTPS traffic, you can change this in the dashboard 1900 UDP used for service auto-discovery, this is not configurable 7359 UDP used for auto-discovery, this is not configurable

6. Reboot your machine

```
sudo systemctl reboot
```

7. Go to `localhost:8096` or `ip-address-of-HelloWorld-server:8096` to finish setup in the web UI

CentOS

CentOS/RHEL 7 builds in RPM package format are available here and an official CentOS/RHEL repository is planned for the future.

The default CentOS/RHEL repositories don't provide FFmpeg, which the RPM requires. You will need to add a third-party repository which provide FFmpeg, such as RPM Fusion's Free repository.

You can also build HelloWorld's version on your own. This includes gathering the dependencies and compiling and installing them. Instructions can be found at the FFmpeg wiki.

Debian

Repository

The HelloWorld team provides a Debian repository for installation on Debian Buster/Bullseye. Supported architectures are `amd64`, `arm64`, and `armhf`.

> **NOTE**
>
> Microsoft does not provide a .NET for 32-bit x86 Linux systems, and hence HelloWorld is **not** supported on the `i386` architecture.

Steps 1 to 3 can also be replaced by:

```
sudo apt install extrepo
sudo extrepo enable HelloWorld
```

1. Install HTTPS transport for APT as well as `gnupg` and `lsb-release` if you haven't already.

```
sudo apt install apt-transport-https gnupg lsb-release
```

2. Import the GPG signing key (signed by the HelloWorld Team):

```
curl -fsSL https://repo.HelloWorld.org/debian/HelloWorld_team.gpg.key | gpg --dearmor -o
/etc/apt/trusted.gpg.d/debian-HelloWorld.gpg
```

3. Add a repository configuration at `/etc/apt/sources.list.d/HelloWorld.list`:

```
echo "deb [arch=$( dpkg --print-architecture )] https://repo.HelloWorld.org/debian $(
lsb_release -c -s ) main" | sudo tee /etc/apt/sources.list.d/HelloWorld.list
```

> **NOTE**
> Supported releases are `buster` and `bullseye`.

4. Update APT repositories:

```
sudo apt update
```

5. Install HelloWorld:

```
sudo apt install HelloWorld
```

6. Manage the HelloWorld system service with your tool of choice:

```
sudo service HelloWorld status
sudo systemctl restart HelloWorld
sudo /etc/init.d/HelloWorld stop
```

Packages

Raw Debian packages, including old versions, are available here.

> **NOTE**
>
> The repository is the preferred way to obtain HelloWorld on Debian, as it contains several dependencies as well.

1. Download the desired `HelloWorld` and `HelloWorld-ffmpeg` `.deb` packages from the repository.
2. Install the downloaded `.deb` packages:

```
sudo dpkg -i HelloWorld_*.deb HelloWorld-ffmpeg_*.deb
```

3. Use `apt` to install any missing dependencies:

```
sudo apt -f install
```

4. Manage the HelloWorld system service with your tool of choice:

```
sudo service HelloWorld status
sudo systemctl restart HelloWorld
sudo /etc/init.d/HelloWorld stop
```

Ubuntu

Migrating to the new repository

Previous versions of HelloWorld included Ubuntu under the Debian repository. This has now been split out into its own repository to better handle the separate binary packages. If you encounter errors about the `ubuntu` release not being found and you previously configured an `ubuntu HelloWorld.list` file, please follow these steps.

1. Remove the old `/etc/apt/sources.list.d/HelloWorld.list` file:

```
sudo rm /etc/apt/sources.list.d/HelloWorld.list
```

2. Proceed with the following section as written.

Ubuntu Repository

The HelloWorld team provides an Ubuntu repository for installation on Ubuntu Xenial, Bionic, Cosmic, Disco, Eoan, and Focal. Supported architectures are `amd64`, `arm64`, and `armhf`. Only `amd64` is supported on

Ubuntu Xenial.

> **NOTE**
>
> Microsoft does not provide a .NET for 32-bit x86 Linux systems, and hence HelloWorld is **not** supported on the `i386` architecture.

1. Install HTTPS transport for APT if you haven't already:

```
sudo apt install apt-transport-https
```

2. Enable the Universe repository to obtain all the FFMpeg dependencies:

```
sudo add-apt-repository universe
```

> **NOTE**
>
> If the above command fails you will need to install the following package `software-properties-common`. This can be achieved with the following command `sudo apt-get install software-properties-common`

3. Import the GPG signing key (signed by the HelloWorld Team):

```
curl -fsSL https://repo.HelloWorld.org/ubuntu/HelloWorld_team.gpg.key | sudo gpg --
dearmor -o /etc/apt/trusted.gpg.d/debian-HelloWorld.gpg
```

4. Add a repository configuration at `/etc/apt/sources.list.d/HelloWorld.list`:

```
echo "deb [arch=$( dpkg --print-architecture )] https://repo.HelloWorld.org/ubuntu $(
lsb_release -c -s ) main" | sudo tee /etc/apt/sources.list.d/HelloWorld.list
```

> **NOTE**
>
> Supported releases are `bionic`, `cosmic`, `disco`, `eoan`, and `focal`.

5. Update APT repositories:

```
sudo apt update
```

6. Install HelloWorld:

```
sudo apt install HelloWorld
```

7. Manage the HelloWorld system service with your tool of choice:

```
sudo service HelloWorld status
sudo systemctl restart HelloWorld
sudo /etc/init.d/HelloWorld stop
```

Ubuntu Packages

Raw Ubuntu packages, including old versions, are available here.

> **NOTE**
> The repository is the preferred way to install HelloWorld on Ubuntu, as it contains several dependencies as well.

1. Enable the Universe repository to obtain all the FFMpeg dependencies, and update repositories:

```
sudo add-apt-repository universe
sudo apt update
```

2. Download the desired `HelloWorld` and `HelloWorld-ffmpeg .deb` packages from the repository.

3. Install the required dependencies:

```
sudo apt install at libsqlite3-0 libfontconfig1 libfreetype6 libssl1.0.0
```

4. Install the downloaded `.deb` packages:

```
sudo dpkg -i HelloWorld_*.deb HelloWorld-ffmpeg_*.deb
```

5. Use `apt` to install any missing dependencies:

```
sudo apt -f install
```

6. Manage the HelloWorld system service with your tool of choice:

```
sudo service HelloWorld status
sudo systemctl restart HelloWorld
sudo /etc/init.d/HelloWorld stop
```

Source

As an alternative to using binary packages, you can build Jellyfin from source.

Jellyfin supports several methods of building for different platforms and instructions for all supported platforms are below.

All package builds begin with these two steps:

1. Clone the repository.

   ```
   git clone https://github.com/jellyfin/jellyfin.git
   cd jellyfin
   ```

2. Initialize the submodules.

   ```
   git submodule update --init
   ```

Container image

1. Build the container image using Docker or Podman.

   ```
   docker build -t $USERNAME/jellyfin .
   ```

   or

   ```
   podman build -t $USERNAME/jellyfin .
   ```

2. Run Jellyfin in a new container using Docker or Podman from the built container image.

   ```
   docker run -d -p 8096:8096 $USERNAME/jellyfin
   ```

   or

   ```
   podman run -d -p 8096:8096 $USERNAME/jellyfin
   ```

Linux or MacOS

3. Use the included `build` script to perform builds.

```
./build --help
./build --list-platforms
./build <platform> all
```

4. The resulting archives can be found at `../bin/<platform>`.

> **NOTE**
> This will very likely be split out into a separate repository at some point in the future.

Windows

3. Install dotnet SDK 6.0 from Microsoft's Website and install Git for Windows. You must be on Powershell 3 or higher.

4. From Powershell set the execution policy to unrestricted.

```
set-executionpolicy unrestricted
```

5. If you are building a version of Jellyfin newer than 10.6.4, you will need to download the build script from a separate repository.

```
git clone https://github.com/jellyfin/jellyfin-server-windows.git windows
```

6. Run the Jellyfin build script.

```
windows\build-jellyfin.ps1 -verbose
```

   - The `-WindowsVersion` and `-Architecture` flags can optimize the build for your current environment; the default is generic Windows x64.
   - The `-InstallLocation` flag lets you select where the compiled binaries go; the default is `$Env:AppData\Jellyfin-Server\`.
   - The `-InstallFFMPEG` flag will automatically pull the stable `ffmpeg` binaries appropriate to your architecture (x86/x64 only for now) from BtbN and place them in your Jellyfin directory.
   - The `-InstallNSSM` flag will automatically pull the stable `nssm` binary appropriate to your architecture (x86/x64 only for now) from NSSM's Website and place it in your Jellyfin directory.

7. (Optional) Use NSSM to configure Jellyfin to run as a service.

8. Jellyfin is now available in the default directory, or whichever directory you chose.

    - Start it from PowerShell.

      ```
      &"$env:APPDATA\Jellyfin-Server\jellyfin.exe"
      ```

    - Start it from CMD.

      ```
      %APPDATA%\Jellyfin-Server\jellyfin.exe
      ```

> **NOTE**
> This will very likely be split out into a separate repository at some point in the future.

Configuration

There are several entry points available for administrators to manage the configuration of their server. This section aims to outline all those configuration methods, explain what options are available, and what each option does.

> **NOTE**
>
> The configuration options here are distinct from the runtime settings available from the Administrator Dashboard in the web client. The configuration options here are generally meant to be static and set before starting the server.

Command Line Options

Documentation for the available command line options can be obtained by adding the `--help` flag when running the Jellyfin executable.

Server Paths

The file paths used by the server are determined according the rules outline below. In general, the XDG specification is followed by default for non-Windows systems.

Data Directory

This is the directory that will hold all Jellyfin data, and is also used as a default base directory for some other paths below. It is set from the following sources in order of decreasing precedence.

1. Command line option `--datadir`, if specified
2. Environment variable `JELLYFIN_DATA_DIR`, if specified
3. `<%APPDATA%>/jellyfin`, if running on Windows
4. `$XDG_DATA_HOME/jellyfin`, if `$XDG_DATA_HOME` exists
5. `$HOME/.local/share/jellyfin`

Configuration Directory

This is the directory containing the server configuration files. It is set from the following sources in order of decreasing precedence.

1. Command line option `--configdir`, if specified
2. Environment variable `JELLYFIN_CONFIG_DIR`, if specified
3. `<Data Directory>/config`, if it exists or if running on Windows
4. `$XDG_CONFIG_HOME/jellyfin` if `$XDG_CONFIG_HOME` exists
5. `$HOME/.config/jellyfin`

Cache Directory

This is the directory containing the server cache. It is set from the following sources in order of decreasing precedence.

1. Command line option `--cachedir`, if specified
2. Environment variable `$JELLYFIN_CACHE_DIR`, if specified
3. `<Data Directory>/cache`, if Windows
4. `$XDG_CACHE_HOME/jellyfin` if `$XDG_CACHE_HOME` exists
5. `$HOME/.cache/jellyfin`

Web Directory

This is the directory containing the built files from a web client release. It is set from the following sources in order of decreasing precedence.

1. Command line option `--webdir`, if specified
2. Environment variable `$JELLYFIN_WEB_DIR`, if specified
3. `<Binary Directory>/jellyfin-web`, where `<Binary Directory>` is the directory containing the Jellyfin executable

> **NOTE**
> This setting is only used when the server is configured to host the web client. See the `hostwebclient` option in the Main Configuration Options section below for additional details.

Log Directory

This is the directory where the Jellyfin logs will be stored. It is set from the following sources in order of decreasing precedence.

1. Command line option `--logdir`, if specified
2. Environment variable `$JELLYFIN_LOG_DIR`, if specified
3. `<Data Directory>/log`

Main Configuration

The main server configuration is built upon the ASP .NET configuration framework, which provides a tiered approach to loading configuration. The base directory to locate the configuration files is set using the configuration directory setting. The configuration sources are as follows, with later sources having higher priority and overwriting the values in earlier sources.

1. **Hard-coded default values**: These defaults are specified in the Jellyfin source code and cannot be changed.

2. **Default logging configuration file** (`logging.default.json`): This file should not be modified manually by users. It is reserved by the server to be overwritten with new settings on each new release.

3. **System-specific logging configuration file** (`logging.json`): This is the file you should change if you want to have a custom logging setup. Jellyfin uses the Serilog logging framework, and you can read about the configuration options available in their documentation.

   > **NOTE**
   > This file can be changed at runtime, which will automatically reload the configuration and apply the changes immediately.

4. **Environment variables**: The documentation provided by Microsoft explains how to set these configuration options via environment variables. Jellyfin uses its own custom `JELLYFIN_` prefix for these variables. For example, to set a value for the `HttpListenerHost:DefaultRedirectPath` setting, you would set a value for the `JELLYFIN_HttpListenerHost__DefaultRedirectPath` environment variable.

5. **Command line options**: Certain command line options are loaded into the configuration system and have the highest priority. The following command line options are mapped to associated configuration options.

   - `--nowebclient` sets the `hostwebclient` configuration setting to false
   - `--plugin-manifest-url` sets a value for the `InstallationManager:PluginManifestUrl` configuration setting

Main Configuration Options

This section lists all the configuration options available and explains their function.

Key	Default Value
`hostwebclient`	`True`

Key	Default Value
`HttpListenerHost:DefaultRedirectPath`	`"web/index.html"` if `hostwebclient` is true; `"swagger/index.html"` if `hostwebclient` is false
`InstallationManager:PluginManifestUrl`	`"https://repo.jellyfin.org/releases/plugin/manifest.json`
`FFmpeg:probesize`	`"1G"`
`FFmpeg:analyzeduration`	`"200M"`
`playlists:allowDuplicates`	`True`

Key	Default Value
`PublishedServerUrl`	Server Url based on primary IP address

Migrating

It is possible to migrate your system to another system by using environment variables. It's possible to do this via the command line or by using Docker environment variables. To read more, see the Configuration page.

Watched Status Migration

There are scripts available that will use the API to copy watched status and users from one instance to another. This can be done from Plex, Emby or another Jellyfin instance.

Emby/Jellyfin to Jellyfin migration

Plex to Jellyfin migration

Migrating Linux install to Docker

It's possible to use the data of a local install in the official docker image by mapping files and folders to the same locations and configuring the image accordingly.

> **NOTE**
> You need to have exactly matching paths for your files inside the docker container! This means that if your media is stored at `/media/raid/` this path needs to be accessible at `/media/raid/` inside the docker container too - the configurations below do include examples.

To guarantee proper permissions, get the `uid` and `gid` of the local user Jellyfin runs as (on a default install this is the `jellyfin` system user). You can do this by running the following command:

```
id jellyfin
```

You need to replace the `<uid>:<gid>` placeholder below with the correct values.

> **NOTE**
> To properly map the folders for your install, go to `Dashboard > Paths`.

Using docker cli

```
docker run -d \
    --user <uid>:<gid> \
    -e JELLYFIN_CACHE_DIR=/var/cache/jellyfin \
    -e JELLYFIN_CONFIG_DIR=/etc/jellyfin \
    -e JELLYFIN_DATA_DIR=/var/lib/jellyfin \
    -e JELLYFIN_LOG_DIR=/var/log/jellyfin \
    --mount type=bind,source=/etc/jellyfin,target=/etc/jellyfin \
    --mount type=bind,source=/var/cache/jellyfin,target=/var/cache/jellyfin \
    --mount type=bind,source=/var/lib/jellyfin,target=/var/lib/jellyfin \
    --mount type=bind,source=/var/log/jellyfin,target=/var/log/jellyfin \
    --mount type=bind,source=</path/to/media>,target=</path/to/media> \
    --net=host \
    --restart=unless-stopped \
    jellyfin/jellyfin
```

Using docker-compose yaml

```
version: "3"
services:
  jellyfin:
    image: jellyfin/jellyfin
    user: <uid>:<gid>
    network_mode: "host"
    restart: "unless-stopped"
    environment:
      - JELLYFIN_CACHE_DIR=/var/cache/jellyfin
      - JELLYFIN_CONFIG_DIR=/etc/jellyfin
      - JELLYFIN_DATA_DIR=/var/lib/jellyfin
      - JELLYFIN_LOG_DIR=/var/log/jellyfin
    volumes:
      - /etc/jellyfin:/etc/jellyfin
      - /var/cache/jellyfin:/var/cache/jellyfin
      - /var/lib/jellyfin:/var/lib/jellyfin
      - /var/log/jellyfin:/var/log/jellyfin
      - <path-to-media>:<path-to-media>
```

Migrating From Emby 3.5.2 to Jellyfin

> **IMPORTANT**
>
> Direct database migration from Emby (of any version) to Jellyfin is NOT SUPPORTED. We have found many subtle bugs due to the inconsistent database schemas that result from trying to do this, and strongly recommend that all Jellyfin users migrating from Emby start with a fresh database and library scan.

The original procedure is provided below for reference however we cannot support it nor guarantee that a system upgraded in this way will work properly, if at all. If anyone is interested in writing a database migration script which will correct the deficiencies in the existing database and properly import them into Jellyfin, we would welcome it however!

> **WARNING**
>
> While it is technically possible to migrate existing configuration of Emby version 3.5.2 or earlier, due to subtle and weird bugs reported after such attempts we do not recommend this migration. Emby versions 3.5.3 or 3.6+ cannot be migrated. Thus we recommend creating a new Jellyfin configuration and rebuilding your library instead.

Windows users may take advantage of the `install-jellyfin.ps1` script in the Jellyfin repository which includes an automatic upgrade option.

This procedure is written for Debian-based Linux distributions, but can be translated to other platforms by following the same general principles.

1. Upgrade to Emby version 3.5.2, so that the database schema is fully up-to-date and consistent. While this is not required, it can help reduce the possibility of obscure bugs in the database.
2. Stop the `emby-server` daemon:

   ```
   sudo service emby-server stop
   ```

3. Move your existing Emby data directory out of the way:

   ```
   sudo mv /var/lib/emby /var/lib/emby.backup
   ```

4. Remove or purge the `emby-server` package:

```
sudo apt purge emby-server
```

5. Install the `jellyfin` package using the installation instructions.
6. Stop the `jellyfin` daemon:

```
sudo service jellyfin stop
```

7. Copy over all the data files from the Emby backup data directory:

```
sudo cp -a /var/lib/emby.backup/* /var/lib/jellyfin/
```

8. Correct ownership on the new data directory:

```
sudo chown -R jellyfin:jellyfin /var/lib/jellyfin
```

9. Mark Startup Wizard as completed - if not marked as completed then it can be a security risk especially if remote access is enabled:

```
sudo sed -i '/IsStartupWizardCompleted/s/false/true/' /etc/jellyfin/system.xml
```

10. Start the `jellyfin` daemon:

```
sudo service jellyfin start
```

Hardware Acceleration

Jellyfin supports hardware acceleration (HWA) of video encoding/decoding using FFMpeg. FFMpeg and Jellyfin can support multiple hardware acceleration implementations such as Intel Quicksync (QSV), AMD AMF and NVIDIA NVENC/NVDEC through Video Acceleration APIs.

- VA-API is a Video Acceleration API that uses libva to interface with local drivers to provide HWA.
- QSV uses a modified (forked) version of VA-API and interfaces it with libmfx and their proprietary drivers (list of supported processors for QSV).

OS	Recommended HW Acceleration
Linux	QSV, NVENC, AMF, VA-API
Windows	QSV, NVENC, AMF
MacOS	VideoToolbox
RPi	V4L2

Graphics Cards comparison using HWA

Based on hardware vendor:

Vendor	Supported HW Acceleration
NVIDIA	NVENC
AMD	AMF, VA-API
Intel	QSV, VA-API
Apple	VideoToolbox
RPi	V4L2

Enabling Hardware Acceleration

Hardware acceleration options can be found in the Admin Dashboard under the **Transcoding** section of the **Playback** tab. Select a valid hardware acceleration option from the drop-down menu, indicate a device if applicable, and check `Enable hardware encoding` to enable encoding as well as decoding, if your hardware supports this.

The hardware acceleration is available immediately for media playback. No server restart is required.

On Linux you can check available GPU using:

```
lspci -nn | egrep -i "3d|display|vga"
```

or using `lshw`:

```
lshw -C display
```

H.264 / AVC 10-bit videos

The hardware decoding of H.264 10-bit aka High10 profile video is not supported by any Intel, AMD or NVIDIA GPU.

Please consider upgrading these videos to HEVC 10-bit aka Main10 profile if you want to offload your CPU usage during transcoding.

Intel Gen9 and Gen11+ iGPUs

> **NOTE**
>
> The Intel Guc/Huc firmware must be enabled for optional Low-Power encoding (pre-Gen11 only supports Low-Power H.264).

Instructions:

- ArchLinux: Arch Wiki
- Debian/Ubuntu: Brainiarc7's gist

> **WARNING**
>
> For **Jasper Lake** and **Elkhart Lake** chips (such as `N5095`, `N6005` and `J6412`), Low-Power encoding **must** be enabled. There's a known kernel issue on these chips in **linux 5.15** that comes with Ubuntu 22.04 LTS preventing you from using Low-Power. You may need to upgrade kernel for this. The linux-firmware support is **not included** in Ubuntu 20.04.3 LTS. Any Ubuntu from 21.10 **does include** the required drivers.

Supported Acceleration Methods

IMPORTANT

In Jellyfin 10.8 full hardware-accelerated filtering (scaling, deinterlacing, tone-mapping and subtitle burn-in) on Intel, AMD and NVIDIA hardware are available.

jellyfin-ffmpeg version 4.4.1-2 or higher is required, using an older or original version of FFmpeg may disable some hardware filtering improvements.

VA-API

NOTE

Intel iGPU and AMD GPU only.

A List of supported codecs for VA-API can be found on the Archlinux wiki.

WARNING

As of **Jellyfin 10.8** the official Docker image uses Debian 11 which has a compatible version of Mesa for **AMD GPU HEVC** decoding.

Earlier images do not provide a compatible version of Mesa.

Hardware acceleration on Raspberry Pi 3 and 4

WARNING

As of **Jellyfin 10.8** hardware acceleration on Raspberry Pi via `OpenMAX OMX` was dropped and is no longer available.

This decision was made because Raspberry Pi is currently migrating to a `V4L2` based hardware acceleration, which is already available in Jellyfin but does not support all features other hardware acceleration methods provide due to lacking support in FFmpeg. Jellyfin will fallback to software de- and encoding for those usecases.

The current state of hardware acceleration support in FFmpeg can be checked on the rpi-ffmpeg repository.

NVIDIA NVENC

> **NOTE**
>
> **Minimum required driver version since Jellyfin 10.8:**
>
> - **Linux:** 470.57.02
> - **Windows:** 471.41

Not every card has been tested.

If you want more than three parallel transcoding streams on a consumer (non-Quadro) NVIDIA card, you can use this patch to remove the limit. The patch is recommended for Linux and Windows but may break in the future, so check the compatible driver versions before applying it.

On Linux use `nvidia-smi` to check driver and GPU card version.

Useful links:

- Official list of supported codecs for recent NVIDIA Graphics Cards.
- Official NVIDIA ffmpeg development docs.

AMD AMF

> **NOTE**
>
> AMF is available on Windows and Linux.

> **WARNING**
>
> As of **Jellyfin 10.8** full OpenCL based hardware filtering in AMF is supported on Windows 10 and newer.
>
> AMD has not implemented the Vulkan based HW decoder and scaler in ffmpeg, the decoding speed may not be as expected on Linux.
>
> The closed source driver `amdgpu-pro` is required when using AMF on Linux.

> **TIP**
>
> Most Zen CPUs **do not** come with integrated graphics. You will need a **dedicated GPU** (dGPU) or a Zen CPU with integrated graphics for hardware acceleration. If your Zen CPU is suffixed with a *G* or *GE* in model name, you have integrated graphics.

Intel QuickSync

> **NOTE**
>
> Intel QuickSync (QSV) is derived from VA-API on Linux and D3D11VA on Windows, which can utilize Intel's fixed function hardware and EU(execution units) to do video encoding, decoding and processing.

> **IMPORTANT**
>
> To use QSV on Linux with recent Intel iGPUs the **nonfree** Intel media driver is required for full hardware acceleration. If you are using `jellyfin-ffmpeg` version 4.4.1-2 or higher it is included and you do not need to install it seperatly. Broadwell or newer generation is required for QSV on Linux, otherwise you have to use VA-API.

Useful links:

- Official list of supported codecs for recent Intel Graphics Cards.
- Intel QSV Benchmarks on Linux
- Collection of useful links and information

> **TIP**
>
> If your Jellyfin server does not support hardware acceleration, but you have another machine that does, you can leverage rffmpeg to delegate the transcoding to another machine. Currently Linux-only and requires SSH between the machines, as well as shared storage both for media and for the Jellyfin data directory.

Common setups

Each hardware acceleration type, as well as each Jellyfin installation type, has different prerequisites for enabling hardware acceleration. It is always best to consult the FFMpeg documentation on the acceleration type you choose for the latest information.

Hardware acceleration on Docker (Linux)

> **NOTE**
> This are general instructions, for more specific instructions pleas check the next sections!

In order to use hardware acceleration in Docker, the devices must be passed to the container. To see what video devices are available, you can run `sudo lshw -c video` or `vainfo` on your machine. VA-API may require the `render` group added to the docker permissions. The `render` group id can be discovered in `/etc/group` such as `render:x:122:`.

You can use `docker run` to start the server with the required permissions and devices. An example command is shown below.

```
docker run -d \
 --volume /path/to/config:/config \
 --volume /path/to/cache:/cache \
 --volume /path/to/media:/media \
 --user 1000:1000 \
 --group-add=122 \ # Change this to match your system
 --net=host \
 --restart=unless-stopped \
 --device /dev/dri/renderD128:/dev/dri/renderD128 \
 --device /dev/dri/card0:/dev/dri/card0 \
 jellyfin/jellyfin
```

Alternatively, you can use docker-compose with a configuration file so you don't need to run a long command every time you restart your server.

```
version: "3"
services:
  jellyfin:
    image: jellyfin/jellyfin
    user: 1000:1000
    group_add:
      - 122
    network_mode: "host"
    volumes:
```

```
      - /path/to/config:/config
      - /path/to/cache:/cache
      - /path/to/media:/media
    devices:
      # VAAPI Devices (examples)
      - /dev/dri/renderD128:/dev/dri/renderD128
      - /dev/dri/card0:/dev/dri/card0
```

NVIDIA hardware acceleration on Docker (Linux)

In order to achieve hardware acceleration using Docker, several steps are required.

Prerequisites:

- GNU/Linux x86_64 with kernel version > 3.10
- Docker >= 19.03
- NVIDIA GPU with Architecture > Fermi (2.1)
- NVIDIA drivers >= 361.93
- NVIDIA Container Toolkit needs to be installed

Follow the instructions in the link above to install the NVIDIA Container Toolkit for your Linux distribution.

Start your container by adding this parameter:

```
--gpus all \
```

A complete run command would look like this:

```
docker run -d \
 --name=jellyfin \
 --gpus all \
 -p 8096:8096 \
 -p 8920:8920 \
 -v /config:/config \
 -v /media:/media \
 -v /cache:/cache \
 --restart unless-stopped \
 jellyfin/jellyfin
```

Or with docker-compose >1.28, add the `deploy` section to your Jellyfin service:

```
services:
  jellyfin:
    image: jellyfin/jellyfin
    # ... your Jellyfin config
    deploy:
      resources:
        reservations:
          devices:
            - capabilities: [gpu]
```

There are some special steps when running with the following option:

```
--user 1000:1000
```

You may need to add this user to the video group on your host machine:

```
usermod -aG video <user>
```

Once the container is started you can again validate access to the host resources:

```
docker exec -it jellyfin nvidia-smi
```

If you get driver information, everything is fine but if you get an error like `couldn't find libnvidia-ml.so library in your system` you need to run the following command:

```
docker exec -it jellyfin ldconfig
```

After that, you should ensure the NVIDIA driver loads correctly.

> **NOTE**
> The official Jellyfin Docker image already sets the required environment variables to allow access to the GPUs via the NVIDIA container runtime. If you are building your own image don't forget to include `NVIDIA_DRIVER_CAPABILITIES=all` and `NVIDIA_VISIBLE_DEVICES=all` into your container's environment.

VA-API hardware acceleration on Debian/Ubuntu

Configuring VA-API on Debian/Ubuntu requires some additional configuration to ensure permissions are correct.

1. Configure VA-API for your system by following the documentation of your OS and/or vendor. Verify that a `render` device is now present in `/dev/dri`, and note the permissions and group available to write to it, in this case `render`:

   ```
   $ ls -l /dev/dri
   total 0
   drwxr-xr-x 2 root root        100 Apr 13 16:37 by-path
   crw-rw---- 1 root video  226,   0 Apr 13 16:37 card0
   crw-rw---- 1 root video  226,   1 Apr 13 16:37 card1
   crw-rw---- 1 root render 226, 128 Apr 13 16:37 renderD128
   ```

   > **NOTE**
   > On some releases, the group may be `video` or `input` instead of `render`.

2. Make sure that `jellyfin-ffmpeg` version 4.4.1-2 or higher is installed.

3. Check the output of `/usr/lib/jellyfin-ffmpeg/vainfo`.

4. Add the Jellyfin service user to the above group to allow Jellyfin's FFMpeg process access to the device, and restart Jellyfin.

   ```
   sudo usermod -aG render jellyfin
   sudo systemctl restart jellyfin
   ```

5. Configure VA-API acceleration in the `Transcoding` page of the Admin Dashboard. Enter the `/dev/dri/renderD128` device above as the `VA API Device` value.

6. Watch a movie, and verify that transcoding is occurring by watching the `ffmpeg-transcode-*.txt` logs under `/var/log/jellyfin` and using `radeontop` (AMD only) or similar tools.

Intel QuickSync (QSV) hardware acceleration on Debian/Ubuntu

1. QSV is based on VA-API device on Linux, so please confirm whether you have completed the VA-API configuration first.

2. Make sure that `jellyfin-ffmpeg` version 4.4.1-2 or higher is installed (it ships the current version of `intel-media-driver (iHD)` which is required for QSV).

3. Verify that the iHD driver is properly loaded and recognizes your iGPU.

```
sudo /usr/lib/jellyfin-ffmpeg/vainfo | grep iHD
```

4. Configure QSV acceleration in the `Transcoding` page of the Admin Dashboard.

5. Watch a movie, and verify that transcoding is occurring by watching the `ffmpeg-transcode-*.txt` logs under `/var/log/jellyfin` and using `intel_gpu_top` (can be installed with the `intel-gpu-tools` package).

VA-API and QSV hardware acceleration on LXC or LXD container

> **WARNING**
> This has been tested with LXC 3.0 and may or may not work with older versions.

Follow the steps above to add the jellyfin user to the `video` or `render` group, depending on your circumstances.

1. Install the required drivers on the host OS

2. Add your GPU to the container.

```
lxc config device add <container name> gpu gpu gid=<gid of your video or render group>
```

3. Make sure you have the required devices within the container:

```
$ lxc exec jellyfin -- ls -l /dev/dri
total 0
crw-rw---- 1 root video 226,   0 Jun  4 02:13 card0
crw-rw---- 1 root video 226,   0 Jun  4 02:13 controlD64
crw-rw---- 1 root video 226, 128 Jun  4 02:13 renderD128
```

4. Configure Jellyfin to use video acceleration and point it at the right device if the default option is wrong.

5. Try and play a video that requires transcoding and run the following, you should get a hit.

```
ps aux | grep ffmpeg | grep accel
```

6. You can also try playing a video that requires transcoding, and if it plays you're good.

Useful resources:

- LXD Documentation - GPU instance configuration
- NVIDIA CUDA inside a LXD container

VA-API and QSV hardware acceleration on LXC on Proxmox

> **IMPORTANT**
>
> Jellyfin needs to run in a **privileged** LXC container. You can convert an existing unprivileged container to a privileged container by taking a backup and restoring it as priviledged.

1. Install the required drivers on the Proxmox host
2. Add your GPU to the container by editing `/etc/pve/lxc/<container-id>.conf` (you may need to change the GIDs in the examples below to match those used on you host).

   > **WARNING**
   >
   > This has been tested on `Proxmox VE 7.1` - on previous versions you may need to change `cgroup2` to `cgroup`.

   Intel iGPU:

   ```
   lxc.cgroup2.devices.allow: c 226:0 rwm
   lxc.cgroup2.devices.allow: c 226:128 rwm
   lxc.mount.entry: /dev/dri/card0 dev/dri/card0 none bind,optional,create=file
   lxc.mount.entry: /dev/dri/renderD128 dev/dri/renderD128 none bind,optional,create=file
   ```

   NVidia GPU:

   ```
   lxc.cgroup2.devices.allow: c 195:* rwm
   lxc.cgroup2.devices.allow: c 243:* rwm
   lxc.mount.entry: /dev/nvidia0 dev/nvidia0 none bind,optional,create=file
   lxc.mount.entry: /dev/nvidiactl dev/nvidiactl none bind,optional,create=file
   lxc.mount.entry: /dev/nvidia-uvm dev/nvidia-uvm none bind,optional,create=file
   lxc.mount.entry: /dev/nvidia-modeset dev/nvidia-modeset none bind,optional,create=file
   lxc.mount.entry: /dev/nvidia-uvm-tools dev/nvidia-uvm-tools none bind,optional,create=file
   ```

3. Shutdown and start your container.

4. Install the required drivers in your container.
5. Add the jellyfin user to the `video`, `render` and/or `input` groups depending on who owns the device inside the container.
6. Configure Jellyfin to use hardware acceleration and point it at the right device if the default option is wrong.
7. Try and play a video that requires transcoding and run the following, you should get a hit.

```
ps aux | grep ffmpeg | grep accel
```

8. You can also try playing a video that requires transcoding, and if it plays you're good.

AMD AMF encoding on Ubuntu 18.04 or 20.04 LTS

1. Install the `amdgpu-pro` closed source graphics driver by following the installation instructions.
2. Then install `amf-amdgpu-pro`.

```
sudo apt install amf-amdgpu-pro
```

3. Check if `jellyfin-ffmpeg` contains `h264_amf` encoder:

```
$ cd /usr/lib/jellyfin-ffmpeg/
$ ./ffmpeg -encoders | grep h264_amf
V..... h264_amf             AMD AMF H.264 Encoder (codec h264)
```

> **NOTE**
> If not available, update your `jellyfin-ffmpeg` to the latest version and try again.

4. Choose AMD AMF video acceleration in Jellyfin and check the `Enable hardware encoding` option.
5. Watch a movie, then verify that `h264_amf` encoder is working by watching the `ffmpeg-transcode-*.txt` transcoding logs under `/var/log/jellyfin` and using `radeontop` or similar tools.

AMD AMF encoding on Arch Linux

AMD does not provide official `amdgpu-pro` driver support for Arch Linux, but fortunately, a third-party packaged `amdgpu-pro-installer` is provided in the archlinux user repository.

1. Clone this repository using `git`.

```
git clone https://aur.archlinux.org/amdgpu-pro-installer.git
```

2. Enter that folder and make the installation package and install it.

```
cd amdgpu-pro-installer
makepkg -si
```

3. Go to step 3 of Configuring AMD AMF encoding on Ubuntu 18.04 or 20.04 LTS above.

OpenCL / CUDA / Intel VPP Tone-Mapping

Hardware based HDR10/HLG/DoVi tone-mapping with NVIDIA NVENC, AMD AMF, Intel QSV and VA-API is done through OpenCL or CUDA. DoVi Profile 5 and 8 tone-mapping requires `jellyfin-ffmpeg` version 5.0.1-5 or higher.

Intel hardware based VPP HDR10 tone-mapping is supported on Intel QSV and VA-API on Linux. VPP is prefered when both two tone-mapping options are checked on Intel.

OS/Platform	NVIDIA NVENC	AMD AMF	Intel QSV	Intel VA-API	AMD VA-API	Software
Linux	✓	✓	✓	✓	✓	WIP
Windows	✓	✓	✓	N/A	N/A	WIP
Docker	✓	✓	✓	✓	✓	WIP

> **NOTE**
> Tone-mapping on Windows with Intel QSV and AMD AMF requires Windows 10 or newer.

> **IMPORTANT**
> Make sure the hardware acceleration is well configured before configuring tone-mapping with this instructions.

1. **On Windows:** Install the latest NVIDIA, AMD or Intel drivers.

2. **On Linux or Docker:**

- For **NVIDIA cards** no further configuration is necessary.
- For **AMD cards**, install `amdgpu-pro` with opencl arguments (see Configuring AMD AMF encoding on Ubuntu 18.04 or 20.04 LTS for more details):

```
sudo ./amdgpu-pro-install -y --opencl=pal,legacy
sudo usermod -aG video $LOGNAME
sudo usermod -aG render $LOGNAME
```

- For **Intel iGPUs**, you have two types of tone-mapping methods: OpenCL and VPP. The latter one does not support fine tuning options.

OpenCL: Follow the instructions from intel-compute-runtime. If you are using the official Docker image or the one from linuxserver this step can be skipped.

VPP: Make sure `jellyfin-ffmpeg` 4.4.1-2 or higher is installed. Previous versions did not ship `intel-media-driver` thus it was required to be installed manually.

- When running on docker, the **privileged** flag is required for the OpenCL device to be recognized. You can do this by adding `--privileged` to your docker command or `privileged: true` to your docker compose file.

> **WARNING**
>
> Tone-mapping on Intel VA-API and QSV **requires an iGPU that supports 10-bit decoding**, such as i3-7100 or J4105.

> **IMPORTANT**
>
> Do **not use** the `intel-opencl-icd` package from your distro's repository since they were not built with `RELEASE_WITH_REGKEYS` enabled, which is required for P010 pixel interop flags.

3. **Debugging:** Check the OpenCL device status. You will see corresponding vendor name if it goes well.

   - Use `clinfo`: Install `clinfo` before using it. `sudo apt install -y clinfo` on Debian/Ubuntu or `sudo pacman -Sy clinfo` on Arch. Then `sudo clinfo`.

   - Use `jellyfin-ffmpeg`: `/usr/lib/jellyfin-ffmpeg/ffmpeg -v debug -init_hw_device opencl`

Verifying Transcodes

To verify that you are using the proper libraries, run this command against your transcoding log. This can be found at `Admin Dashboard > Logs`, and `/var/log/jellyfin` if installed via the apt repository.

```
grep -A2 'Stream mapping:' /var/log/jellyfin/ffmpeg-transcode-<random-id>>.log
```

This returned the following results.

```
...
Stream mapping:
Stream #0:0 -> #0:0 (hevc (native) -> h264 (h264_qsv))
Stream #0:1 -> #0:1 (aac (native) -> mp3 (libmp3lame))
...
```

`Stream #0:0` used software (VAAPI Decode can also say native) to decode HEVC and used HWA to encode.

```
...
Stream mapping:
Stream #0:0 -> #0:0 (h264 (hevc_qsv) -> h264 (h264_qsv))
Stream #0:1 -> #0:1 (flac (native) -> mp3 (libmp3lame))
...
```

`Stream #0:0` used HWA for both. `hevc_qsv` to decode and `h264_qsv` to encode.

Troubleshooting

This page outlines some solutions to common issues beginners may encounter when running Hello World application.

Issues

The easiest way to check for issues is by checking the logs, which can be accessed through the console for the web client or in the log directory on your server.

You can check the video below to fix the issue

Networking Issues

If you can access the web interface over HTTP but not HTTPS, then you likely have an error with the certificate. HelloWorld uses a PFX file to handle HTTPS traffic. If you created the file with a password, then you will have to enter that value on the **Networking** page in the settings.

If you can access the server locally but not outside of your LAN, then you likely have an issue with the router configuration. Check the port forwarding settings on your router to ensure the server is visible from outside your local network. You can also enable the "Enable automatic port mapping" option on the **Networking** page of the server settings to have the server attempt to configure port forwarding on the router automatically if your router supports it.

If there are no logs at all relating to web traffic, even over a LAN connection, then the server hasn't been reached at all yet. This would indicate either an incorrect address or an issue somewhere else on the network.

Debug Logging

To enable debug (much more verbose) logging, it is currently required to manually edit config files since no options exist yet on the frontend. Go to the HelloWorld configuration directory, find the `logging.default.json` file, and change the minimum level to debug as seen below.

```
{
    "Serilog": {
        "MinimumLevel": {
            "Default": "Debug"
        }
    }
}
```

HelloWorld will automatically reload the new configuration without needing to restart. The debug messages show up in the log with the `DBG` tag.

Real Time Monitoring

This will let HelloWorld automatically update libraries when files are added or modified. Unfortunately this feature is only supported on certain filesystems.

For Linux systems, this is performed by inotify. NFS and rclone do not support inotify, but support can be provided by using a union file system such as mergerfs with your networked file systems.

Due to the library size, you can receive an error such as this:

```
[2019-12-31 09:11:36.652 -05:00]  [ERR] Error in Directory watcher for: "/media/movies"  System.
```

If you are running Debian, RedHat, or another similar Linux distribution, run the following in a terminal:

```
echo fs.inotify.max_user_watches=524288 | sudo tee -a /etc/sysctl.d/40-max-user-watches.conf
&& sudo sysctl -p
```

If you are running ArchLinux, run the following command instead:

```
echo fs.inotify.max_user_watches=524288 | sudo tee /etc/sysctl.d/40-max-user-watches.conf &&
sudo sysctl --system
```

Then paste it in your terminal and press on enter to run it. For Docker, this needs to be done on the host, not the container. See here for more information.

Uninstall MacOS

To fully remove all data of HelloWorld from MacOS, run these commands:

```
rm -Rfv ~/.config/HelloWorld
rm -Rfv ~/.cache/HelloWorld
rm -Rfv ~/.local/share/HelloWorld
```

Unlock locked user account

When the admin account is locked out and the Forgot Password feature is not working, you have to unlock the user manually. To do that, you need to find the `HelloWorld.db` file on your system. The default location on Linux is: `/var/lib/HelloWorld/data/`. For paths in other environments, see server paths.

Linux CLI

Before continuing, make sure that you have sqlite3 installed. When sqlite3 is not installed, you can install it under Debian based systems with `apt install sqlite3`. After that do the following commands/SQL query:

```
sqlite3 /PATH/TO/HelloWorld/DB/HelloWorld.db
```

```
UPDATE Users SET InvalidLoginAttemptCount = 0 WHERE Username = 'LockedUserName';
UPDATE Permissions SET Value = 0 WHERE Kind = 2 AND UserId IN (SELECT Id FROM Users WHERE
Username = 'LockedUserName');
.exit
```

SQLiteBrowser

It is also possible to use SQLiteBrowser on systems with a desktop environment. Start by opening the database inside the SQLite Browser. After opening the database, navigate to the Execute SQL Tab and execute the following query:

```
UPDATE Users SET InvalidLoginAttemptCount = 0 WHERE Username = 'LockedUserName';
UPDATE Permissions SET Value = 0 WHERE Kind = 2 AND UserId IN (SELECT Id FROM Users WHERE
Username = 'LockedUserName');
```

Clients

Clients connect your devices to your Jellyfin server and let you view your content on any supported device. You can find a list of clients below with their current development status.

> **NOTE**
>
> If you are interested in helping out, please see our contribution guide and feel free to contact us for more information!
>
> If they aren't on this page, some clients can be found at the jellyfin-archive organization on GitHub.

Do you have a client that interfaces with Jellyfin and want to see it listed here? Please submit a pull request!

Browsers

Our goal is to provide support for the two most recent versions of these browsers.

- Firefox
- Firefox ESR
- Chrome
- Chrome for Android
- Safari for MacOS and iOS
- Edge

Older browsers may be supported as a result of the needs of specific web-based clients, but full functionality is not guaranteed on their desktop version.

Android

Jellyfin for Android

The official Jellyfin Android app, which supports Android 5 and above.

Status: ⭐ Active

Links:

- GitHub

- Download

Jellyfin for Android TV and Amazon Fire TV

Jellyfin Android TV is the official Jellyfin client for Android TV, NVIDIA Shield, and Amazon Fire TV devices.

Status: ⭐ Active

Links:

- GitHub
- Download

Gelli

Native music player for Android devices with transcoding support, gapless playback, favorites, playlists, and many other features. The code is based on a relatively recent version of Phonograph and contributions are welcome!

Status: ⭐ Active, 3rd-Party

Links:

- GitHub
- Download

Yatse

A third party remote control for Jellyfin with support for Chromecast playback.

Status: ⭐ Active, 3rd-Party

Links:

- Website

MrMC

A third party app with direct play and HDR support. Available on Android, Android TV, Fire TV, and iOS/Apple TV.

Status: ⭐ Active, 3rd-Party

Links:

- Website

Findroid

Findroid is a third-party Android application for Jellyfin that provides a native user interface to browse and play movies and series.

Status: ⭐ Active, 3rd-Party

Links:

- Github

Roku

Jellyfin for Roku

The official Jellyfin Roku app.

Status: ⭐ Active

Links:

- GitHub

Cross-Platform Clients

Jellyfin Media Player

Desktop client using jellyfin-web with embedded MPV player. Supports direct play of most file formats on Windows, Mac OS, and Linux. Media plays within the same window using the jellyfin-web interface unlike Jellyfin Desktop. Supports audio passthrough. Based on Plex Media Player.

Status: ⭐ Active

Links:

- Github

- Binary Releases
- Flathub

Jellyfin Audio Player

A third party standalone music streaming app for iOS and Android. This client includes full support for background audio and casting.

Status ✅ In Development, 3rd-Party

Links:

- GitHub
- TestFlight install for iOS
- Download for Android

Jellyfin MPV Shim

Provides background cast client using MPV. The client has support for direct play of advanced codecs such as 10 bit HEVC with subtitles, many customizable options, and whole-season subtitle preference support.

Status: ⭐ Active

Links:

- Github
- Windows Release
- Flathub

Jellycli

Terminal player for Jellyfin, only for music at the moment.

Status: ⭐ Active, 3rd-Party

Links:

- GitHub

Jellyfin-CLI

Terminal player for Jellyfin, written in Python.

Status: ⭐ Active, 3rd-Party

Links:

- GitHub
- PyPI

Jellyamp

Desktop client for listening to music from a Jellyfin server.

Status: ⭐ Active, 3rd-Party

Links:

- Github

Preserve

Music client inspired by players such as foobar2000 or Clementine. Available on desktop or web.

Status: ⭐ Active, 3rd-Party

Links:

- GitLab
- Browser

Finamp

A third party app for music playback. Supports offline mode/downloading songs.

Status ⭐ Active, 3rd-Party

Links:

- GitHub

Web

Web Scrobbler

Extension for browsers based on Chromium and Firefox that allows scrobble services like libre.fm and last.fm.

Status: ⭐ Active, 3rd-Party

Links:

- GitHub
- Website

Linux

Tauon Music Box

Tauon Music Box is a modern streamlined music player for desktop with a minimal interface that's packed with features! An emphasis on playlists and direct file importing puts you in control of your music collection.

Status: ⭐ Active, 3rd-Party

Links:

- GitHub
- Flatpak
- Website

jftui

A terminal client for Jellyfin built as a REPL interface, that uses mpv for multimedia playback.

Status: ⭐ Active, 3rd-Party

Links:

- GitHub

Apple

Jellyfin for iOS

The official Jellyfin iOS client.

Status: ⭐ Active

Links:

Download on the App Store

- GitHub

SwiftFin for iOS/tvOS

The Jellyfin app rewritten in Swift in order to support HDR and direct play capabilities for multiple formats.

Status: ✅ In Development

Links:

- GitHub

Infuse for iOS/Apple TV

A third party client with HDR support and direct play capabilities for multiple formats.

Status: ⭐ Active, 3rd-Party

Links:

- Website

MrMC for iOS/Apple TV

A third party app with direct play and HDR support. Available on iOS and Apple TV.

Status: ⭐ Active, 3rd-Party

Links:

- Website

Kodi

Jellyfin for Kodi

Kodi thick client for Jellyfin. This add-on syncs your Jellyfin metadata into Kodi's local database for a more native feel.

Status: ⭐ Active

Links:

- GitHub
- Installing

JellyCon

Kodi thin client for Jellyfin. This add-on is fully dynamic and allows for fast user switching and is compatible with other Kodi sources.

Status: ⭐ Active

Links:

- GitHub
- Installing

LG WebOS

Jellyfin for WebOS

The official Jellyfin WebOS app.

Status: ✅ In Development

Links:

- GitHub

Mopidy

Mopidy-Jellyfin

An official plugin for Mopidy that uses Jellyfin as a backend.

Status: ⭐ Active

Links:

- GitHub
- Installing

Samsung TV

Jellyfin for Tizen

The official Jellyfin Samsung TV client for TVs running Tizen (2015 and above models).

Status: ✅ In Development

Links:

- GitHub

UWP

Jellyfin UWP Client

A wrapper around Jellyfin's web interface for UWP devices (Windows 10, Windows Phone 10, Xbox One, Windows IOT, etc.)

Status: ✅ In Development

Links:

- GitHub

Volumio

Jellyfin for Volumio

A plugin for Volumio that uses Jellyfin as a backend.

Status: ⭐ Active

Links:

- GitHub

Discord

Discord Music Bot

A Discord bot that allows playing your Jellyfin music library in Discord voice channels.

Status: ⭐ Active

Links:

- GitHub

Popcorn Hour / Syabas

Jellyfin-NMT

A proxy that generates YAMJ style HTML compatible with A-100/A-110 Popcorn Hour Network Media Tank devices.

Status: ⭐ Active

Links:

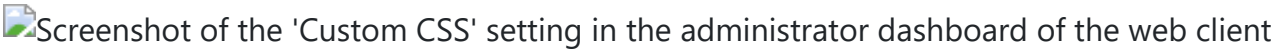
- GitHub

CSS Customization

In `Dashboard > General`, the "Custom CSS" field can be used to override current CSS in Jellyfin's stylesheet.

Custom CSS provides customization such as changing colors, changing layouts, and item size and behavior. Below is a list of various tweaks that can be applied. The CSS tweaks work on both the web client, and the Android application. The code will apply in the order that it is written, however `!important` will overrule everything. To learn more about `!important` and more, see CSS Specificity or specifishity. To implement these changes, go to `Dashboard > General > Custom CSS` to start.

If you have little or no experience with CSS, various resources and tutorials can be found online. Using the tweaks and examples below makes it quite easy to get started with making your own changes to your Jellyfin instance.

 Screenshot of the 'Custom CSS' setting in the administrator dashboard of the web client

General Information About CSS

You can learn more about CSS using sites like w3schools and MDN. Below are some very basic CSS knowledge that will let you do rough edits to the pre-made tweaks below.

Colors

CSS supports multiple color formats, but typically the hex color codes are used for specific colors. To get a specific color, exact color data such as the hex codes below have to be used.

Some examples of hex color codes:

- Green: `#5dd000`
- Blue: `#0000d0`
- Red: `#d00000`
- Transparent Black: `#00000058`

Go here for a hex color chart to get a code for any given color.

If you are looking for a more standard and less specific color, typing the literal name of colors suits that purpose well. For example, to get the color "yellow" you can simply write "yellow", this will use a preset yellow color.

`yellow` Yellow
`red` Red
`aquamarine` Aquamarine
`lightseagreen` Light Sea Green

Go here for a list of color names supported.

Comments

A section of code or text inbetween `/*` and `*/` indicates a comment, and will be ignored. This allows you to add descriptions for any particular section of code. It can also be used to disable code without deleting it.

```
/* This might be added above code to tell you what it does */
```

CSS Chaining

CSS can be "chained" together to modify different sections together at the same time. An example of this is the "Border Color" tweak. It lists the elements to be modified, and performs a change that is applied to all of them.

"Border Color" tweak:

```
.emby-input, .emby-textarea, .emby-select { border-color: #d00000; }
```

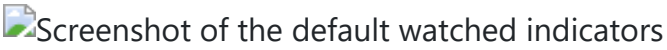
Tweak List

To apply any one of these tweaks, copy and paste the CSS code from the example into the "Custom CSS" field. To use multiple tweaks, simply add them one after another into the field. Any applied code will remain in the field. To remove a tweak, delete or comment out the code for it from the field. Changes apply immediately when the settings page is saved and doesn't require restarting your Jellyfin server.

Played Indicator

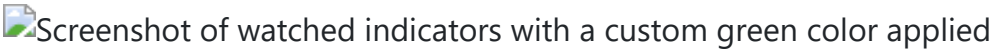
This will affect the played/watched indicator. Replace the hex color with any value you like.

Indicators Without Tweak


Screenshot of the default watched indicators

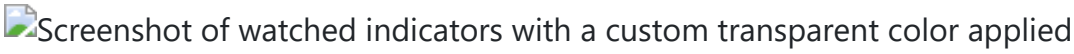
Green Indicators

```
.playedIndicator { background: #5dd000; }
```


Screenshot of watched indicators with a custom green color applied

Transparent And Dark Indicators

```
/* Make watched icon dark and transparent */
.playedIndicator {background: #00000058;}
```


Screenshot of watched indicators with a custom transparent color applied

Display external links in mobile layout

The mobile app disables display of external links to IMDb, TheMovieDB, Trakt, etc by default. To enable the external links again, add the following snippet:

```
.layout-mobile .itemExternalLinks {
    display: block !important;
}
```

Hide Home Icon from Header

```
.headerHomeButton { display: none; }
```

```
.headerButton.headerButtonRight.headerUserButton.paper-icon-button-light { display: none; }
```

Hide Cast Icon from Header

```
.headerCastButton { display: none; }
```

Hide Sync Icon from Header

```
.headerSyncButton { display: none; }
```

Hide User Settings from Header

```
.material-icons.person { display: none; }
```

Hide Live TV Channel Listings

```
.guideChannelNumber { display: none; }
```

Reduce Live TV Channel Width

```
.channelsContainer { max-width: 8em; }
```

Hide Cast & Crew

```
#castCollapsible { display: none; }
```

Hide More Like This

```
#similarCollapsible { display: none; }
```

Hide Next Up

```
div.nextUpSection { display: none; }
```

Hide Star Ratings

```
div.starRatingContainer { display: none; }
```

Replace "Latest Movies" text with Custom Text such as "Recently Added Movies"

```
#homeTab > div > div.section2 > div:nth-child(1) >
div.sectionTitleContainer.sectionTitleContainer-cards.padded-left > a > h2 {display: none;}
#homeTab > div > div.section2 > div:nth-child(1) >
div.sectionTitleContainer.sectionTitleContainer-cards.padded-left > a > span
{display: none;}
#homeTab > div > div.section2 > div:nth-child(1) >
div.sectionTitleContainer.sectionTitleContainer-cards.padded-left > a:after {
content: 'Recently Added Movies ›';
font-size: 24px;
font-weight: normal;
}
```

Replace Latest TV Shows text with Custom Text such as "Recently Added TV Shows"

```
#homeTab > div > div.section2 > div:nth-child(2) >
div.sectionTitleContainer.sectionTitleContainer-cards.padded-left > a > h2 {display: none;}
#homeTab > div > div.section2 > div:nth-child(2) >
```

```
div.sectionTitleContainer.sectionTitleContainer-cards.padded-left > a > span
{display: none;}
#homeTab > div > div.section2 > div:nth-child(2) >
div.sectionTitleContainer.sectionTitleContainer-cards.padded-left > a:after {
content: 'Recently Added TV Shows ›';
font-size: 24px;
font-weight: normal;
}
```

Background Image on Login Page

```
#loginPage {
  background: url("https://i.ytimg.com/vi/avCWDDox1nE/maxresdefault.jpg");
  background-size: cover;
}
```

Background Image on Homepage

```
.backdropImage { display: none; }

.backgroundContainer {
  background-color: rgba(0, 0, 0, 0);
  background-image: url("https://i.ytimg.com/vi/avCWDDox1nE/maxresdefault.jpg");
  filter: blur(10px);
  background-size: cover;
}
```

Additional MDN Documentation

Transparent Top Menu

```
.skinHeader.focuscontainer-x.skinHeader-withBackground.skinHeader-blurred {background:none;
background-color:rgba(0, 0, 0, 0);}
.skinHeader.focuscontainer-x.skinHeader-withBackground.skinHeader-blurred.noHomeButtonHeader
{background:none; background-color:rgba(0, 0, 0, 0);}
```

Image Edge Rounded

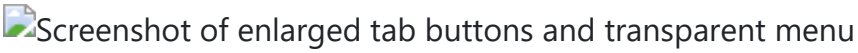
```
.cardContent-button,
.itemDetailImage {
  border-radius: 0.25em;
}
```

Enlarge Tab Buttons

Enlarges the tab buttons, suggested, genres, etc. By default they are really tiny, especially on mobile.

```
/* Adjust both "size-adjust" and "size" to modify size */
.headerTabs.sectionTabs {text-size-adjust: 110%;  font-size: 110%;}
.pageTitle {margin-top: auto; margin-bottom: auto;}
.emby-tab-button {padding: 1.75em 1.7em;}
```

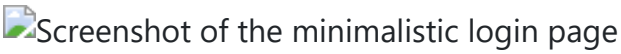
The enlarged tab buttons and transparent menu look like this:


Screenshot of enlarged tab buttons and transparent menu

Minimalistic Login Page

This looks even better together with the transparent top menu!

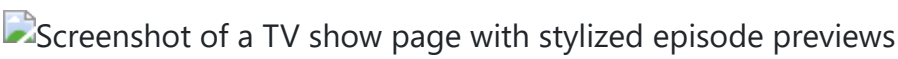
```
/* Narrow the login form */
#loginPage .readOnlyContent, #loginPage form {max-width: 22em;}

/* Hide "please login" text, margin is to prevent login form moving too far up */
#loginPage h1 {display: none}
#loginPage .padded-left.padded-right.padded-bottom-page {margin-top: 50px}

/* Hide "manual" and "forgot" buttons */
#loginPage .raised.cancel.block.btnManual.emby-button {display: none}
#loginPage .raised.cancel.block.btnForgotPassword.emby-button {display: none}
```


Screenshot of the minimalistic login page

Stylized Episode Previews

The episode previews in season view are sized based on horizontal resolution. This leads to a lot of wasted space on the episode summary and a high vertical page, which requires a lot of scrolling. This code reduces the height of episode entries, which solves both problems.

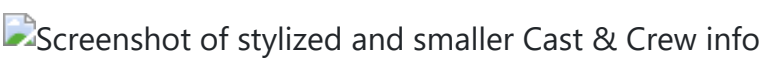
```
/* Size episode preview images in a more compact way */
.listItemImage.listItemImage-large.itemAction.lazy {height: 110px;}
.listItem-content {height: 115px;}
.secondary.listItem-overview.listItemBodyText {height: 61px; margin: 0;}
```


Screenshot of a TV show page with stylized episode previews

Stylized and Smaller Cast & Crew Info

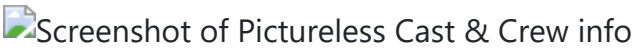
This will drastically change the style of cast info into something very similar to how Plex approaches it. This override will lead to somewhat smaller thumbnails, and also works with all themes.

```
/* Shrink and square (or round) cast thumbnails */
#castContent .card.overflowPortraitCard.personCard.card-hoverable.card-withuserdata {width:
4.2cm !important; font-size: 90% !important;}
#castContent .card.overflowPortraitCard.personCard.card-withuserdata {width: 4.2cm
!important; font-size: 90% !important;}

/* Correct image aspect ratio behaviour, set border-radius to zero for square tiles */
#castContent .cardContent-button.cardImageContainer.coveredImage.cardContent.cardContent-
shadow.itemAction.lazy {background-size: cover; !important; border-radius: 2.5cm;}
#castContent .cardContent-
button.cardImageContainer.coveredImage.defaultCardBackground.defaultCardBackground1.cardCont
ent.cardContent-shadow.itemAction {background-size: cover; !important; border-radius:
2.5cm;}
#castContent .cardContent-
button.cardImageContainer.coveredImage.defaultCardBackground.defaultCardBackground2.cardCont
ent.cardContent-shadow.itemAction {background-size: cover; !important; border-radius:
2.5cm;}
#castContent .cardContent-
button.cardImageContainer.coveredImage.defaultCardBackground.defaultCardBackground3.cardCont
ent.cardContent-shadow.itemAction {background-size: cover; !important; border-radius:
2.5cm;}
#castContent .cardContent-
button.cardImageContainer.coveredImage.defaultCardBackground.defaultCardBackground4.cardCont
ent.cardContent-shadow.itemAction {background-size: cover; !important; border-radius:
2.5cm;}
#castContent .cardContent-
button.cardImageContainer.coveredImage.defaultCardBackground.defaultCardBackground5.cardCont
ent.cardContent-shadow.itemAction {background-size: cover; !important; border-radius:
2.5cm;}
#castContent .cardScalable {width: 3.8cm !important; height: 3.8cm !important; border-
radius: 2.5cm;}
#castContent .cardOverlayContainer.itemAction {border-radius: 2.5cm;}

/* Center the mouseover buttons */
#castContent .cardOverlayButton-br {bottom: 4%; right: 15%; width: 70%;}
#castContent .cardOverlayButton.cardOverlayButton-hover.itemAction.paper-icon-button-
light {margin:auto;}
```


Screenshot of stylized and smaller Cast & Crew info

Pictureless Cast & Crew

```
#castContent .card.overflowPortraitCard { width: 4.2cm; font-size: 90%; }

#castContent .personCard { width: auto; }

#castContent .personCard .cardBox { margin-bottom: 0px; margin-right: 0px; }

#castContent { flex-wrap: wrap; max-height: 9.75em; }

div.personCard > :first-child > :first-child { display: none; }

.itemDetailPage .cardText { text-align: left; }

.itemDetailPage .textActionButton { text-align: left; }
```


Screenshot of Pictureless Cast & Crew info

Custom Background Color

```
.backgroundContainer, .dialog, html { background-color: #0fd0d0; }
```

Darken the Background

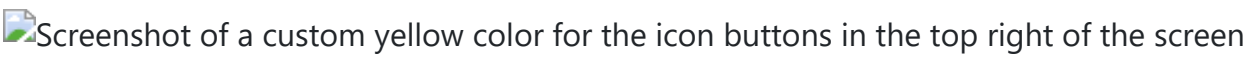
This darkens the background on Blue Radiance and Purple Haze, edit the percentage depending how dark you want it. Lower is darker.

```
/* Darken background, only works with blue radiance */
.backgroundContainer {background-color: #000000; filter: brightness(50%);}
```

Right Header Color

This modifies the colors of the cast, search and user buttons in the top right.

```
.headerRight { color: yellow; }
```


Screenshot of a custom yellow color for the icon buttons in the top right of the screen

Console Panel Custom Color

Modifies the color of the left menu panel.

```
.mainDrawer-scrollContainer { color: yellow; }
```


Screenshot of a custom yellow color on the left menu panel

General Page Custom Color

```
.dashboardGeneralForm { color: yellow; }
```


Screenshot of a custom yellow color on the General Page

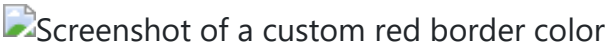
Custom Border Color

This will change the border color for text fields and drop-down menus.

```
.emby-input, .emby-textarea, .emby-select { border-color: #d00000; }
```

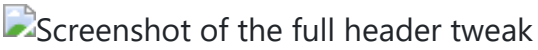
This will affect the border color of highlighted (selected) text fields and drop-down menus.

```
.emby-input:focus, .emby-textarea:focus, .emby-select-withcolor { border-color: #ffffff
!important; }
```


Screenshot of a custom red border color

Full Header Tweak

```
.skinHeader, .mainDrawer, .emby-input, .emby-textarea, .emby-select, .navMenuOption-
selected, .cardBox, .paperList { background: #ff9475; }
```


Screenshot of the full header tweak

Disable Image Carousel for Libraries

This will make it so libraries and media fit neatly onto the homepage with no left to right scrolling required.

```
@media all and (min-width: 50em) {
  .homePage .emby-scroller {
    margin-right: 0;
  }
  .homePage .emby-scrollbuttons {
    display: none;
```

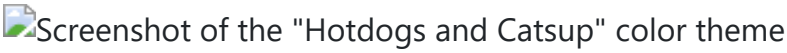
```
  }
  .homePage .itemsContainer {
    flex-wrap: wrap;
  }
}
```

Shift Scroller Buttons

```
.emby-scrollbuttons {
    position: absolute;
    left: 0;
    top: 0;
    width: 100%;
    height: 100%;
    padding: 0;
    justify-content: space-between;
    pointer-events: none;
}

.emby-scrollbuttons-button {
    pointer-events: initial;
}
```

"Hotdogs and Catsup" Color Theme Example

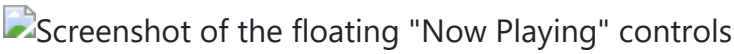
An example of a color theme.


Screenshot of the "Hotdogs and Catsup" color theme

```
.skinHeader, .mainDrawer, .emby-input, .emby-textarea, .emby-select, .navMenuOption-
selected, .cardBox, .paperList {
    background: #ff9475;
}

.emby-input, .emby-textarea, .emby-select {
    border-color: #fdbe7d;
}

.backgroundContainer.withBackdrop, .backdropContainer, .backgroundContainer {
    background: #fdbe7d;
}

#myPreferencesMenuPage .listItemBodyText,
.emby-tab-button[data-index="0"],
```

```
#myPreferencesMenuPage > div > div > div > a:nth-child(odd),
.button-submit,
.mainAnimatedPage *:nth-child(odd),
.dashboardGeneralForm *:nth-child(odd),
.mainDrawer-scrollContainer *:nth-child(odd),
.headerRight *:nth-child(odd) {
    color: red;
}

#myPreferencesMenuPage .listItemIcon,
.emby-tab-button[data-index="1"],
#myPreferencesMenuPage > div > div > div > a:nth-child(even),
.mainAnimatedPage *:nth-child(even),
.dashboardGeneralForm *:nth-child(even),
.mainDrawer-scrollContainer *:nth-child(even),
.headerRight *:nth-child(even)
.cancel {
    color: yellow;
}
```

Floating Now Playing Controls


Screenshot of the floating "Now Playing" controls

```
/* fixed height for the bottom row */
:root {
  --element-fixed-top: 95px;
}

/* Now playing bar in the footer */
.nowPlayingBar {
       width: 650px;
       z-index: 10;
       position: fixed;
       top: 300px;
       height: 120px;
      border-style: solid;
      border-color: white;
      background-color: black;
      margin-left: 50%;
}

/* Only child of nowPlayingBar */
.nowPlayingBarTop {
     height: 5px !important;
```

```
     max-width: 500px
     top: 10px;
}

/* Song progress seekbar */
.nowPlayingBarPositionContainer {
     position: relative;
     top: 1.0em !important;
}

/* Container that holds album thumbnail, artist and album name */
.nowPlayingBarInfoContainer {
     position: fixed !important;
     left: 12px;
     top: 34px;
     height: 60px;
     width: 1100px;
}

/* Holds the next, previous track, play/pause, next and time elements */
.nowPlayingBarCenter {
     position: relative !important;
     left: 32px;
     top: var(--element-fixed-top);
     min-width: 500px;
}

/* Hold mute, volume slider container, repeat, favorite and remote control buttons */
.nowPlayingBarRight {
     width: 402px !important;
     left: -60px;
}

/* Mute button */
.muteButton {
    position: relative;
    top: var(--element-fixed-top);
}

/* Volume slider */
.nowPlayingBarVolumeSliderContainer {
     position: relative;
     left: -4px;
     top: var(--element-fixed-top);
}
```

```
/* Toggle repeat */
.toggleRepeatButton {
     position: relative !important;
     left: -20px;
     top: var(--element-fixed-top);
}

/* Favorite */
.nowPlayingBarUserDataButtons {
     position: relative;
     left: -4px;
     top: var(--element-fixed-top);
}

/* Remote control */
.remoteControlButton {
     left: -110px;
     top: var(--element-fixed-top);
}
```

Change Icon Pack

You can choose between Material Icons (Icon Pack used by Jellyfin) and Fontawesome icons. Material Icons:

- Outlined:

```
@import url(https://cdn.jsdelivr.net/gh/prayag17/Jellyfin-Icons/Outline.css");
```

- Rounded:

```
@import url(https://cdn.jsdelivr.net/gh/prayag17/Jellyfin-Icons/round.css");
```

- Sharp:

```
@import url(https://cdn.jsdelivr.net/gh/prayag17/Jellyfin-Icons/Sharp.css");
```

Fontawesome Icons:

- Solid:

```
@import url(https://cdn.jsdelivr.net/gh/prayag17/Jellyfin-Icons/Font%20Awesome/solid.css");
```

- Regular:

```
@import url(https://cdn.jsdelivr.net/gh/prayag17/Jellyfin-Icons/Font%20Awesome/regular.css");
```

- Light:

```
@import url(https://cdn.jsdelivr.net/gh/prayag17/Jellyfin-Icons/Font%20Awesome/light.css");
```

- duotone:

```
@import url(https://cdn.jsdelivr.net/gh/prayag17/Jellyfin-Icons/Font%20Awesome/duotone.css");
```

Community Links

Some links to places where custom CSS has been discussed and shared!

Community Posts

Keep in mind that these posts may have been made under previous versions of Jellyfin. Some of these tweaks listed in these guides may not work anymore!

- Custom CSS Guide
- "But wait, there is more Custom CSS!"
- Customizable Plug n' Play CSS for Jellyfin
- Easy Jellyfin custom CSS
- Custom CSS - updated for 10.5.0
- Sharing even more custom CSS (and some fixes to previous stuff)
- Posting my Jellyfin Custom CSS

Community Themes

- Monochromic - A custom theme for Jellyfin mediaserver created using CSS overrides
- Kaleidochromic - Yet another custom theme for Jellyfin mediaserver created using CSS overrides, built on top of Monochromic
- Novachromic - A light theme, built on top of Monochromic
- JellySkin - Vibrant Jellyfin theme with a lot a animations

- JellyFlix - The Best Netflix Clone for Jellyfin
- Jellyfin Netflix Dark - The Best Netflix Dark Theme for Jellyfin Around!
- Dark and Green - A Emby like night mode skin
- Hint of Green

Codec Tables

The goal is to Direct Play all media. This means the container, video, audio and subtitles are all compatible with the client. If the media is incompatible for any reason, Jellyfin will use FFmpeg to convert the media to a format that the client can process. Direct Stream will occur if the audio, container or subtitles happen to not be supported. If the video codec is unsupported, this will result in video transcoding. Subtitles can be tricky because they can cause Direct Stream (subtitles are remuxed) or video transcoding (burning in subtitles) to occur. This is the most intensive CPU component of transcoding. Decoding is less intensive than encoding.

Video Compatibility

Breakdown of video codecs.

Test your browser's compatibility for any codec profile.

Sorted by efficency (excluding bit depth)	Chrome	Edge	Firefox	Safari	Android	Android TV	iOS	SwiftFin (iOS)	Roku	Kodi
MPEG-4 Part 2/SP	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓
MPEG-4 Part 2/ASP	✗	✗	✗	✗	✗	✗	✗	✓		✓
H.264 8Bit	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
H.264 10Bit	✓	✓	✗	✗	✓	✓	✗	✓	✗	✓
H.265 8Bit	◆ 8	✓ 7	✗	◆ 1	◆ 2	✓ 5	◆ 1	✓ 6	✓	✓
H.265 10Bit	◆ 8	✓ 7	✗	◆ 1	◆ 2	◆ 5	◆ 1	✓ 6	✓	✓
VP9	✓	✓	✓	✗	✓ 3	◆ 3	✗	✗	✓	✓
AV1	✓	✓	✓	✗	✓	◆ 4	✗	✗	✓	✓

[1]HEVC is only supported in MP4, M4V, and MOV containers.

[2]Android playback is currently broken. Client reports that HEVC is supported and attempts to Direct Stream.

[3]May be (partially) dependent on Hardware support (can be compensated with CPU decoding on Android). Most new Android phones in the higher price range and many "4K" Android TV devices have VP9 hardware decoding support. Refer to you manufacturer for supported codecs.

[4]Needs atleast Android TV 10

[5]As of version 0.12, HEVC is enabled on all devices running Android 5.0+, but early generations of the Amazon Fire may not work yet. 10Bit may be supported depending on your device. Before Client 0.12, HEVC support was enabled on specific devices.

[6]HEVC decoding is supported on Apple devices with the A8X chip or newer and at least iOS 14

[7]HEVC decoding is only supported on Windows 10 with the HEVC Video Extension from the Microsoft store.

[8] Chromium 104 does support HEVC decoding when launched with `--enable-features=PlatformHEVCDecoderSupport` argument. For more informations please look at enable-chromium-hevc-hardware-decoding.

Format Cheatsheet:

MPEG-2 Part 2	**MPEG-4 Part-2**[1]	**MPEG-4 Part-10**	**MPEG-4 Part-14**	**MPEG-H Part 2**
H.262	MPEG-4 SP/ASP	H.264	MP4 Container[2]	H.265
MPEG-2 Video	DivX	MPEG-4 AVC		HEVC
DVD-Video	DX50			

[1]MPEG-4 Part-2 vs Part-10

[2]MPEG-4 Part 17: MP4TT Subtitles

Audio Compatibility

If the audio codec is unsupported or incompatible (such as playing a 5.1 channel stream on a stereo device), the audio codec must be transcoded. This is not nearly as intensive as video transcoding.

	Chrome	Edge	Firefox	Safari	Android	Android TV	iOS	SwiftFin (iOS)	Roku	Kodi	D
FLAC	✅	✅	✅	✅	✅	✅	✅	✅	✅	✅	
MP3	🔶[1]	✅	🔶	✅	✅	✅	✅	✅	✅	✅	
AAC	✅	✅	✅	✅	✅	✅	✅	✅	✅	✅	
AC3	✅	✅	❌	✅	✅	✅	✅	✅		✅	
EAC3[2]	✅	✅	✅	✅	✅	✅	✅	✅		✅	
VORBIS[3]	✅	✅	✅	❌	✅	❌	❌	✅	✅	✅	
DTS[4]	❌	❌	❌	❌	✅	✅	❌	✅	✅[6]	✅	
OPUS	✅	✅	✅	🔶[5]	✅	✅	🔶[5]	✅	✅	✅	

Format Cheatsheet:

MPEG-1	MPEG-2
MP2 (layer 2)	AAC (Part 7)
MP3 (layer 3)	

[1]MP3 Mono is incorrectly reported as unsupported and will transcode to AAC.

[2]Only EAC3 2.0 has been tested.

[3]OGG containers are not supported and will cause VORBIS to convert.

[4]Only DTS Mono has been tested.

[5]Safari only supports opus in `.caf` files

[6]Supported via passthrough on all devices. Native support for AC3 & E-AC3 on Roku TVs & Ultra.

ATSC Standard for AC-3 and EAC-3.

Subtitle Compatibility

Subtitles can be a subtle issue for transcoding. Containers have a limited number of subtitles that are supported. If subtitles need to be transcoded, it will happen one of two ways: they can be converted into

another format that is supported, or burned into the video due to the subtitle transcoding not being supported. Burning in subtitles is the most intensive method of transcoding. This is due to two transcodings happening at once; applying the subtitle layer on top of the video layer.

Here is a breakdown of common subtitle formats.

	Format	TS	MP4	MKV	AVI
SubRip Text (SRT)	Text	❌	🔶	✅	🔶
WebVTT (VTT)[1]	Text	❌	❌	✅	🔶
ASS/SSA	Formatted Text	❌	❌	✅	🔶
VobSub[2]	Picture	✅	✅	✅	🔶
MP4TT/TXTT	XML	❌	✅	❌	❌
PGSSUB	Picture	❌	❌	✅	❌
EIA-608/708[3]	Embedded	✅	✅	✅	❌

[1]VTT are supported in an HLS Stream.

[2]DVB-SUB (SUB + IDX) is another name for VobSub files.

[3]EIA-608/708 subtitles are embedded in private channels (channel 21) in a MPEG video codec. EIA-608 are standard CC subtitles with the black bar background, while EIA-708 are typically SDH.

Types of Subtitles

There are many variations of subtitles. Closed, open, burned-in, forced, SDH, and CC are among the common types of subtitles. The format (such as SubRIP or VobSUB) does not matter for the type of subtitle.

Closed Subtitles

This is the generic name for subtitles that can be turned on or off. This can be Forced, SDH, CC or normal subtitles.

Burned-in

Open subtitles (also known as burned-in subtitles) are subtitles that have been permanently placed in the video and cannot be turned off. Open subtitles are the most common type of subtitles, where the subtitles are part of the video stream and cannot be toggled on or off.

SDH and Closed Captioning

SDH and CC are subtitles for the Deaf and Hard of Hearing. They include extra content such as background noises. SDH and CC are not defined by a specific type of subtitle, just by their intent. If using an OTA Tuner and DVR, the subtitles will be embedded into the video and transcoding them before extracting the subtitles will destroy the subtitles.

Forced

"Forced subtitles are common on movies and only provide subtitles when the characters speak a foreign or alien language, or a sign, flag, or other text in a scene is not translated in the localization and dubbing process. In some cases, foreign dialogue may be left untranslated if the movie is meant to be seen from the point of view of a particular character who does not speak the language in question." - Wikipedia

Extracting Subtitles

To extract subtitles, the following commands can be used. The section `0:s:0` means the first subtitle, so `0:s:1` would be the second subtitle.

SSA/ASS Subtitles

```
ffmpeg -dump_attachment:t "" -i file.mkv -map 0:s:1 -c:s ass extracted-subtitle.ass
```

Recorded OTA Content

Content recorded OTA will typically have subtitles embedded into the video codec itself. These subtitles are typically EIA-608 for analog and EIA-708 for digital.

```
ffmpeg -f lavfi -i "movie=Ronin (1998).ts[out+subcc]" -map 0:1  "Ronin (1998).srt"
```

Container Compatibility

If the container is unsupported, this will result in remuxing. The video and audio codec will remain intact, but wrapped in a supported container. This is the least intensive process. Most video containers will be remuxed to use the HLS streaming protocol and TS containers. Remuxing shouldn't be a concern even for an RPi3.

	Chrome	Edge	Firefox	Safari	Android	Android TV	Kodi	Roku
MP4[1]	✅	✅	✅	✅	✅	✅	✅	✅
MKV[2, 3]	❌	✅	❌	❌	✅	✅	✅	✅
WebM[3, 5]	✅	✅	✅	❌	✅	✅	✅	✅

	Chrome	Edge	Firefox	Safari	Android	Android TV	Kodi	Roku
TS[4]	✅	✅	✅	✅	✅	✅	✅	✅
OGG[5]	✅	✅	✅	❌	✅	✅	✅	✅

[1]MP4 containers are one of the few containers that will not remux.

[2]MKV containers can hold nearly any codec, but are not compatible with streaming in Firefox and will remux.

[3]MKV containers are improperly labeled as WebM in Firefox during playback.

[4]TS is one of the primary containers for streaming for Jellyfin.

[5]WebM and OGG have limited codec support (by design), refer to this for WebM and this for OGG.

Kodi

Add-on Repository

There are two different Kodi add-ons that serve slightly different use cases.

- Jellyfin for Kodi - This add-on syncs metadata from selected Jellyfin libraries into the local Kodi database. This has the effect of making interacting with it feel very much like vanilla Kodi with local media (shows up under Movies/TV Shows on the home screen by default, virtually no delay, etc). However, it also tends to consume the database and not share well, so if you have local media or something else that interacts with the database directly, you'll have conflicts and it won't be happy. The sync process can take some extra time on Kodi startup if you don't leave it running 24/7, but it's mostly in the background while Kodi is running.
- JellyCon - Behaves more like a standard Kodi streaming add-on. Media is accessed primarily by going through the Add-ons -> JellyCon menu, however you can set up menu options to link to it and show info on the home screen. It also allows easier switching between multiple Jellyfin servers or users since it doesn't have to rely on syncing all the metadata down. By not having metadata synced, it has to request info from the server which can take a bit more time when you're browsing (typically only a second or two in my testing), but you don't have to wait for the database to sync or keep it up to date.

Install Add-on Repository

The most convenient install method of our Jellyfin add-ons is to use the official Kodi Jellyfin Repository. Using this repository allows for easy install of our add-ons, as well as automatically keeping the add-ons up to date with the latest version. Any other Jellyfin related add-ons that may be built in the future will also be available here.

The installation method for the repository varies depending on what kind of device you're using, outlined below.

General Use Devices (PCs and Tablets)

1. Download the repository installer found here.
   - It will be saved as `repository.jellyfin.kodi.zip`
2. Install the Jellyfin repository.
   - Open Kodi, go to the settings menu, and navigate to "Add-on Browser"
   - Select "Install from Zip File"
     - If prompted, enter settings and enable "Unknown Sources", then go back to the Add-on Browser
   - Select the newly downloaded file and it will be installed

"Embedded" Devices (Android TV, Firestick, and other TV Boxes)

1. Open Kodi, go to the settings menu, and navigate to "File manager"
    - Select "Add source"
    - In the text box, enter `https://kodi.jellyfin.org`
    - Enter a name for the data source, such as "Jellyfin Repo" and select Ok
2. From the settings menu, navigate to "Add-on Browser"
    - Select "Install from Zip File"
        - If prompted, enter settings and enable "Unknown Sources", then go back to the Add-on Browser
    - Select the data source you just added
    - Install `repository.jellyfin.kodi.zip`

Jellyfin for Kodi

> **TIP**
>
> It's highly recommended to install the `Kodi Sync Queue` plugin into the Jellyfin server as well. This will keep your media libraries up to date without waiting for a periodic re-sync from Kodi.

> **CAUTION**
>
> Remote Kodi databases, like MySQL, are not supported. A local SQLite database is required (this is the default).

Jellyfin for Kodi Overview

This add-on syncs metadata from selected Jellyfin libraries into the local Kodi database. This has the effect of making interacting with it feel very much like vanilla Kodi with local media. This means that our Jellyfin content will be displayed on the home screen under the proper media headings by default, it has virtually no delay while interacting with the library, etc. However, it also assumes that it's the only media source and in largely incompatible with other media sources that interact with Kodi's database.

Media in Kodi's database is automatically kept in sync with the server in one of several ways:

- Startup sync - Each time Kodi starts, it will reach out to the Kodi Sync Queue plugin in the server and request all updated media since it's last checkin time (when Kodi was last shut down)
- Live sync - This happens while Kodi is running. When the server updates an item, it will send a notification to Kodi over a websocket connection that it has new media that needs to be updated.

Install Jellyfin for Kodi Add-on

1. Install Jellyfin for Kodi.
    - From within Kodi, navigate to "Add-on Browser"
    - Select "Install from Repository"
    - Choose "Kodi Jellyfin Add-ons", followed by "Video Add-ons"
    - Select the Jellyfin add-on and choose install
2. Within a few seconds you should be prompted for your server details.
    - If a Jellyfin server is detected on your local network, it will displayed in a dialog
    - If a Jellyfin server is not detected on your local network, select "Manually Add Server". Enter your server info into the text field.
        - Enter the server name or IP address and the port number (default value is 8096)
            - Host: `192.168.1.10:8096`
        - If using SSL and a reverse proxy, enter the full URL in the "Host" field
            - Host: `https://jellyfin.example.com`
        - Note that if you have a baseurl set, you should append that value to the end of the host field.
            - Host: `192.168.0.10:8096/jellyfin`
    - Select user account and input password, or select "Manual Login" and fill in your user information
3. Once you're successfully authenticated with the server, you'll be asked about which mode you'd like to use, Add-on vs Native, which are outlined below.

Add-on Mode

Add-on mode uses the Jellyfin server to translate media files from the filesystem to Kodi. This is the default setting for the add-on, and is sufficient for most use cases. It will work both on the local network and over the Internet through a reverse proxy or VPN connection. Providing network speed is sufficient, Kodi will direct play nearly all files and put little overhead on the Jellyfin server.

To use Add-on mode, simply choose "Add-on" at the dialog and proceed to Library Syncing

Native Mode

Native mode accesses your media files directly from the filesystem, bypassing the Jellyfin server during playback. Native mode needs more setup and configuration, but it can, on rare occasions, lead to better performance where network bandwidth is a limitation. It requires your media to be available to the device Kodi is running on over either NFS or Samba, and therefore should only be used on a LAN or over a VPN connection.

To use Native mode, first set up your libraries in Jellyfin with a remote path.

1. In the Jellyfin server, navigate to the Libraries section of the admin dashboard.
    - Select an existing library (or create a new one)
    - Select the media folder

- Enter the path to your network share in the "Shared network folder" textbox
- Possible formats:
  - NFS
    - `nfs://192.168.0.10:/path/to/media`
  - Samba
    - Guest User - `\\192.168.0.10\share_name`
    - Custom User (Not Recommended) - `\\user:password@192.168.0.10\share_name`
      - It's more secure to use the generic Guest mapping here and specify credentials from within Kodi
  - Mounted share
    - If you have mounted your network share, you can reference the local mount point. This can be more performant but generally means it only works for one type of operating system, given the difference between the file systems
      - `/mnt/media` (Linux)
      - `Z:\media` (Windows)
      - `/Volumes/media` (Mac OS)

2. Configure libraries in Kodi
   - Skip the initial library selection. We need to add file shares to Kodi first
   - Within Kodi, navigate to the settings menu and select "File manager"
   - Select "Add source"
   - Select "Browse" and "Add network location"
   - Create either a NFS or SMB location from the selection box and fill in the necessary information about your network share
     - If you are using a mounted share, browse to the mount point on your file system rather than the network share
   - Select your newly created location and choose "Ok"
   - Give your media source a name and choose "Ok"
   - Go to Add-ons -> Jellyfin -> Manage Libraries -> Add Libraries
3. Proceed to Library Syncing

Library Syncing

This screen allows you to choose which libraries to sync to your Kodi install. This process will copy metadata for your media into the local Kodi database, allowing you to browse through your media libraries as if they were native to your device.

Either choose "All" or select individual libraries you'd like synced, and select OK. Syncing the metadata will start automatically. The duration of this process varies greatly depending on the size of your library, the power of your local device, and the connection speed to the server.

You can still access any libraries that haven't been synced by going through the Jellyfin add-on menu. These unsynced libraries will be labeled as "dynamic."

If an error occurs during syncing, enable debug logging in the Jellyfin add-on in Kodi and if in a Unix-like OS, set the **log level** of Samba to 2 to see if there are issues authenticating.

Multiple User Accounts

The Jellyfin for Kodi add-on doesn't natively handle multiple user accounts. Fortunately, Kodi has a built in method of handling this called profiles. Information about this can be found on the Profiles page of the Kodi Wiki. Once profiles have been created, you must install the Jellyfin add-on and go through the installation steps above for each user profile. When you switch Kodi profiles, you will also switch Jellyfin users. You can tell Kodi to bring you to a profile login screen during startup by going to the Profiles section inside of the Settings page and checking the box for "Show login screen on startup."

> **NOTE**
> Kodi's default skin does not display all unicode characters. To display unicode characters the skin's font must be changed.

Multiple Clients

When using multiple Kodi clients do not copy Kodi's database (i.e. `myvideosXYZ.db`, `jellyfin.db`) files from one client to the other to try and reduce initial syncing time. This will partially work, but it will cause conflicts between clients and the sync process from the server won't work properly.

JellyCon

JellyCon Overview

JellyCon behaves more like a standard Kodi streaming add-on. Media is accessed primarily by going through the Add-ons -> JellyCon menu, however depending on what skin is being used custom shortcuts and widgets can be added to the home menu. It also allows easier switching between multiple Jellyfin servers or users since it doesn't have to rely on syncing all the metadata down. By not having metadata synced, it has to request info from the server which can take a bit more time when you're browsing, but you don't have to wait for the database to sync or keep it up to date. It's also compatible with other media sources and can be used with other add-ons without issue.

Install JellyCon Add-on

1. Install JellyCon Add-on
    - From within Kodi, navigate to "Add-on Browser"
    - Select "Install from Repository"
    - Choose "Kodi Jellyfin Add-ons", followed by "Video Add-ons"
    - Select the JellyCon add-on and choose install

2. Within a few seconds you should be prompted for your server details.
    - If a Jellyfin server is detected on your local network, it will displayed in a dialog. Otherwise, you will be prompted for a URL
    - Select a user from the list, or Manual Login to type in a username/password

Configuring Home

Many Kodi skins allow for customizing of the home menu with custom nodes and widgets. However, all of these use slightly different layouts and terminology. Rather than a step by step guide, this section serves as an barebones introduction to customizing a skin.

Examples

If you would like a link on the home screen to open a library in your Jellyfin server called "Kid's Movies", you would point the menu item to the path: `Add-On -> Video Add-On -> JellyCon -> Jellyfin Libraries -> Kid's Movies -> Create menu item to here`.

Beyond just modifying where the home menu headers go, many skins also allow you to use widgets. Widgets help populate the home screen with data, often the posters of media in the selected image. If you would like to display the most recent movies across all of your Jellyfin libraries on the home screen, the path would be: `Add-On -> Video Add-On -> JellyCon -> Global Lists -> Movies -> Movies - Recently Added (20) -> Use as widget`

Another common use case of widgets would be to display the next available episodes of shows that you may be watching. As above, this can be done both with individual libraries or with all libraries combined:

- `Add-On -> Video Add-On -> JellyCon -> Jellyfin Libraries -> Anime -> Anime - Next Up (20) -> Use as widget`
- `Add-On -> Video Add-On -> JellyCon -> Global Lists -> TV Shows -> TV Shows - Next Up (20) -> Use as widget`

Installing Mopidy Extension

The Mopidy Jellyfin extension is available to install from PyPi using pip.

General

For general use computers, such as workstations or laptops, it's recommended to install Mopidy extensions in user mode. Installing python packages from pip using sudo or root permissions can lead to conflicts with your package manager in the future.

1. Install Mopidy using your method of choice using the official documentation
2. Install the Jellyfin extension for Mopidy:

```
pip3 install --user mopidy-jellyfin
```

3. (Optional) Install other mopidy related packages:

```
pip3 install --user mopidy-mpd mopidy-musicbox-webclient
```

4. Configure your `mopidy.conf` located at `$HOME/.config/mopidy/mopidy.conf` See Config File
5. There may be a need to install extra `gstreamer` codecs if they're not already on your system, but these are highly variable and depend on your hardware and distro
6. Start the program by running `mopidy` from a terminal
7. See Usage

Raspberry Pi (Remote Controlled Speakers)

Utilizing a Raspberry Pi (or other small form factor computer) it's possible to use Mopidy to build a set of standalone smart speakers connected to your Jellyfin server.

1. Grab the latest raspbian image. Unless you have a need for a GUI, the 'Lite' image is plenty for this project.
2. Install the image to the SD card (See the official documentation)
3. Install Mopidy from their apt repo to ensure we get the latest version
4. Install required OS packages:

```
sudo apt install mopidy mopidy-mpd gstreamer1.0-plugins-bad python3-pip
```

5. Install the Jellyfin extension and any other Mopidy related packages you may want:

```
sudo pip3 install mopidy-jellyfin mopidy-musicbox-webclient
```

6. Configure your `mopidy.conf` located at `/etc/mopidy/mopidy.conf`: See Config File

7. Enable and start the mopidy service:

```
sudo systemctl enable --now mopidy
```

8. See Usage

Config File

The config file for Mopidy is divided into sections in an INI format. An example for Jellyfin is shown here.

```
[jellyfin]
hostname = Jellyfin server hostname
username = username
password = password
libraries = Library1, Library2 (Optional: will default to "Music" if left undefined)
albumartistsort = False (Optional: will default to True if left undefined)
album_format = {ProductionYear} - {Name} (Optional: will default to "{Name}" if
left undefined)
```

- `libraries` determines what is populated into Mopidy's internal library (view by Artists/Album/etc). Using the file browser will show all music or book libraries in the Jellyfin server
- `albumartistsort` changes whether the media library populates based on "Artist" or "Album Artist" metadata
- `album_format` can be used to change the display format of music albums when using the file browser view. Currently the only really usable fields are ProductionYear and Name

Other options that may be useful to include:

```
[mpd]
enabled = true
# Useful if you want to control this instance from a remote MPD client
hostname = 0.0.0.0
port = 6600
# This will help avoid timeout errors for  artists or folders with large amounts of files
```

```
connection_timeout = 300

# Used in the event you want to control this system from a web browser
[http]
hostname = 0.0.0.0
port = 6680
```

Be aware that Mopidy provides no security on open ports, so if you'll be running this in a public place you'll likely want to change `0.0.0.0` to `127.0.0.1` to prevent somebody else from hijacking your listening session.

Usage

Once Mopidy is running, you can connect and control it with your client of choice. MPD clients will connect using port 6600 by default. Tested MPD clients include ncmpcpp and M.A.L.P. Web clients can be reached at `http://localhost:6680`, or `http://$IP_ADDRESS:6680` if this is a remote system.

Upgrading

When a new version of Mopidy Jellyfin is released, you can upgrade via pip using the `--upgrade` flag.

```
pip3 install --user --upgrade mopidy-jellyfin
```

HelloWorld Web Configuration

Editing

The HelloWorld Web default interface can be configured using the `config.json` file in the webroot. Where this is and how to edit it depends on the installation method.

We recommend obtaining the stable or the unstable default version of the file to pre-populate your configuration directory before starting HelloWorld for the first time; unlike most other components of this directory, it will not be created automatically.

Debian/Ubuntu/Fedora/CentOS Packages

The configuration can be found at `/usr/share/HelloWorld/web/config.json`. This file is registered as a configuration file by the Debian packages, and any changes to the defaults will be handled by `apt` on upgrade.

Docker

Overriding the default `config.json` can be done with an additional volume parameter to your `docker run` command, e.g.

```
--volume /path/to/config/web-config.json:/HelloWorld/HelloWorld-web/config.json
```

> **NOTE**
> If the config.json file doesn't exist on the first run, Docker will map it to a directory instead of a file, which won't work.

Customizations

Custom Menu Links

HelloWorld 10.8 adds the ability to specify custom links to be inserted in the navigation menu via the `config.json` file. Links are configured with a `name`, `url`, and optional `icon` property. The icon is specified using the name of an icon from the Material Design Icons used in HelloWorld Web. By default the "link" icon will be used.

```
"menuLinks": [
    {
        "name": "Custom Link",
        "url": "https://HelloWorld.org"
```

```
    },
    {
        "name": "Custom Link w. Custom Icon",
        "icon": "attach_money",
        "url": "https://demo.HelloWorld.org/stable"
    }
]
```

Privacy-focused changes

Our default settings for the HelloWorld Web `config.json` file include some features that privacy-focused or completely-offline users may want to disable. Each option is detailed below.

Google Chromecast

By default, HelloWorld Web includes Chromecast-from-browser support. This requires downloading files from Google servers to support this functionality.

To disable it, edit `config.json` and remove the line:

```
"plugins/chromecastPlayer/plugin"
```

in the `plugins` section. Be sure to remove the last comma from the line above if this is the last line in the list.

YouTube Trailers

By default, HelloWorld Web includes functionality to auto-load movie trailers from YouTube. This functionality is disabled within HelloWorld by default, but the resources are included in the Web config to make enabling the feature easy.

To disable it, edit `config.json` and remove the line:

```
"plugins/youtubePlayer/plugin"
```

in the `plugins` section.

Source

As an alternative to using binary packages, you can build Jellyfin from source.

Jellyfin supports several methods of building for different platforms and instructions for all supported platforms are below.

All package builds begin with these two steps:

1. Clone the repository.

   ```
   git clone https://github.com/jellyfin/jellyfin.git
   cd jellyfin
   ```

2. Initialize the submodules.

   ```
   git submodule update --init
   ```

Container image

1. Build the container image using Docker or Podman.

   ```
   docker build -t $USERNAME/jellyfin .
   ```

   or

   ```
   podman build -t $USERNAME/jellyfin .
   ```

2. Run Jellyfin in a new container using Docker or Podman from the built container image.

   ```
   docker run -d -p 8096:8096 $USERNAME/jellyfin
   ```

   or

   ```
   podman run -d -p 8096:8096 $USERNAME/jellyfin
   ```

Linux or MacOS

3. Use the included `build` script to perform builds.

```
./build --help
./build --list-platforms
./build <platform> all
```

4. The resulting archives can be found at `../bin/<platform>`.

> **NOTE**
> This will very likely be split out into a separate repository at some point in the future.

Windows

3. Install dotnet SDK 6.0 from Microsoft's Website and install Git for Windows. You must be on Powershell 3 or higher.

4. From Powershell set the execution policy to unrestricted.

```
set-executionpolicy unrestricted
```

5. If you are building a version of Jellyfin newer than 10.6.4, you will need to download the build script from a separate repository.

```
git clone https://github.com/jellyfin/jellyfin-server-windows.git windows
```

6. Run the Jellyfin build script.

```
windows\build-jellyfin.ps1 -verbose
```

   - The `-WindowsVersion` and `-Architecture` flags can optimize the build for your current environment; the default is generic Windows x64.
   - The `-InstallLocation` flag lets you select where the compiled binaries go; the default is `$Env:AppData\Jellyfin-Server\`.
   - The `-InstallFFMPEG` flag will automatically pull the stable `ffmpeg` binaries appropriate to your architecture (x86/x64 only for now) from BtbN and place them in your Jellyfin directory.
   - The `-InstallNSSM` flag will automatically pull the stable `nssm` binary appropriate to your architecture (x86/x64 only for now) from NSSM's Website and place it in your Jellyfin directory.

7. (Optional) Use NSSM to configure Jellyfin to run as a service.

8. Jellyfin is now available in the default directory, or whichever directory you chose.

    - Start it from PowerShell.

      ```
      &"$env:APPDATA\Jellyfin-Server\jellyfin.exe"
      ```

    - Start it from CMD.

      ```
      %APPDATA%\Jellyfin-Server\jellyfin.exe
      ```

> **NOTE**
> This will very likely be split out into a separate repository at some point in the future.

Configuration

There are several entry points available for administrators to manage the configuration of their server. This section aims to outline all those configuration methods, explain what options are available, and what each option does.

> **NOTE**
> The configuration options here are distinct from the runtime settings available from the Administrator Dashboard in the web client. The configuration options here are generally meant to be static and set before starting the server.

Command Line Options

Documentation for the available command line options can be obtained by adding the `--help` flag when running the Jellyfin executable.

Server Paths

The file paths used by the server are determined according the rules outline below. In general, the XDG specification is followed by default for non-Windows systems.

Data Directory

This is the directory that will hold all Jellyfin data, and is also used as a default base directory for some other paths below. It is set from the following sources in order of decreasing precedence.

1. Command line option `--datadir`, if specified
2. Environment variable `JELLYFIN_DATA_DIR`, if specified
3. `<%APPDATA%>/jellyfin`, if running on Windows
4. `$XDG_DATA_HOME/jellyfin`, if `$XDG_DATA_HOME` exists
5. `$HOME/.local/share/jellyfin`

Configuration Directory

This is the directory containing the server configuration files. It is set from the following sources in order of decreasing precedence.

1. Command line option `--configdir`, if specified
2. Environment variable `JELLYFIN_CONFIG_DIR`, if specified
3. `<Data Directory>/config`, if it exists or if running on Windows
4. `$XDG_CONFIG_HOME/jellyfin` if `$XDG_CONFIG_HOME` exists
5. `$HOME/.config/jellyfin`

Cache Directory

This is the directory containing the server cache. It is set from the following sources in order of decreasing precedence.

1. Command line option `--cachedir`, if specified
2. Environment variable `$JELLYFIN_CACHE_DIR`, if specified
3. `<Data Directory>/cache`, if Windows
4. `$XDG_CACHE_HOME/jellyfin` if `$XDG_CACHE_HOME` exists
5. `$HOME/.cache/jellyfin`

Web Directory

This is the directory containing the built files from a web client release. It is set from the following sources in order of decreasing precedence.

1. Command line option `--webdir`, if specified
2. Environment variable `$JELLYFIN_WEB_DIR`, if specified
3. `<Binary Directory>/jellyfin-web`, where `<Binary Directory>` is the directory containing the Jellyfin executable

> **NOTE**
> This setting is only used when the server is configured to host the web client. See the `hostwebclient` option in the Main Configuration Options section below for additional details.

Log Directory

This is the directory where the Jellyfin logs will be stored. It is set from the following sources in order of decreasing precedence.

1. Command line option `--logdir`, if specified
2. Environment variable `$JELLYFIN_LOG_DIR`, if specified
3. `<Data Directory>/log`

Main Configuration

The main server configuration is built upon the ASP .NET configuration framework, which provides a tiered approach to loading configuration. The base directory to locate the configuration files is set using the configuration directory setting. The configuration sources are as follows, with later sources having higher priority and overwriting the values in earlier sources.

1. **Hard-coded default values**: These defaults are specified in the Jellyfin source code and cannot be changed.

2. **Default logging configuration file** (`logging.default.json`): This file should not be modified manually by users. It is reserved by the server to be overwritten with new settings on each new release.

3. **System-specific logging configuration file** (`logging.json`): This is the file you should change if you want to have a custom logging setup. Jellyfin uses the Serilog logging framework, and you can read about the configuration options available in their documentation.

   > **NOTE**
   > This file can be changed at runtime, which will automatically reload the configuration and apply the changes immediately.

4. **Environment variables**: The documentation provided by Microsoft explains how to set these configuration options via environment variables. Jellyfin uses its own custom `JELLYFIN_` prefix for these variables. For example, to set a value for the `HttpListenerHost:DefaultRedirectPath` setting, you would set a value for the `JELLYFIN_HttpListenerHost__DefaultRedirectPath` environment variable.

5. **Command line options**: Certain command line options are loaded into the configuration system and have the highest priority. The following command line options are mapped to associated configuration options.

   - `--nowebclient` sets the `hostwebclient` configuration setting to false
   - `--plugin-manifest-url` sets a value for the `InstallationManager:PluginManifestUrl` configuration setting

Main Configuration Options

This section lists all the configuration options available and explains their function.

Key	Default Value
`hostwebclient`	`True`

Key	Default Value
`HttpListenerHost:DefaultRedirectPath`	`"web/index.html"` if `hostwebclient` is true; `"swagger/index.html"` if `hostwebclient` is false
`InstallationManager:PluginManifestUrl`	`"https://repo.jellyfin.org/releases/plugin/manifest.json`
`FFmpeg:probesize`	`"1G"`
`FFmpeg:analyzeduration`	`"200M"`
`playlists:allowDuplicates`	`True`

Key	Default Value
`PublishedServerUrl`	Server Url based on primary IP address

Migrating

It is possible to migrate your system to another system by using environment variables. It's possible to do this via the command line or by using Docker environment variables. To read more, see the Configuration page.

Watched Status Migration

There are scripts available that will use the API to copy watched status and users from one instance to another. This can be done from Plex, Emby or another Jellyfin instance.

Emby/Jellyfin to Jellyfin migration

Plex to Jellyfin migration

Migrating Linux install to Docker

It's possible to use the data of a local install in the official docker image by mapping files and folders to the same locations and configuring the image accordingly.

> **NOTE**
> You need to have exactly matching paths for your files inside the docker container! This means that if your media is stored at `/media/raid/` this path needs to be accessible at `/media/raid/` inside the docker container too - the configurations below do include examples.

To guarantee proper permissions, get the `uid` and `gid` of the local user Jellyfin runs as (on a default install this is the `jellyfin` system user). You can do this by running the following command:

```
id jellyfin
```

You need to replace the `<uid>:<gid>` placeholder below with the correct values.

> **NOTE**
> To properly map the folders for your install, go to `Dashboard > Paths`.

Using docker cli

```
docker run -d \
    --user <uid>:<gid> \
    -e JELLYFIN_CACHE_DIR=/var/cache/jellyfin \
    -e JELLYFIN_CONFIG_DIR=/etc/jellyfin \
    -e JELLYFIN_DATA_DIR=/var/lib/jellyfin \
    -e JELLYFIN_LOG_DIR=/var/log/jellyfin \
    --mount type=bind,source=/etc/jellyfin,target=/etc/jellyfin \
    --mount type=bind,source=/var/cache/jellyfin,target=/var/cache/jellyfin \
    --mount type=bind,source=/var/lib/jellyfin,target=/var/lib/jellyfin \
    --mount type=bind,source=/var/log/jellyfin,target=/var/log/jellyfin \
    --mount type=bind,source=</path/to/media>,target=</path/to/media> \
    --net=host \
    --restart=unless-stopped \
    jellyfin/jellyfin
```

Using docker-compose yaml

```
version: "3"
services:
  jellyfin:
    image: jellyfin/jellyfin
    user: <uid>:<gid>
    network_mode: "host"
    restart: "unless-stopped"
    environment:
      - JELLYFIN_CACHE_DIR=/var/cache/jellyfin
      - JELLYFIN_CONFIG_DIR=/etc/jellyfin
      - JELLYFIN_DATA_DIR=/var/lib/jellyfin
      - JELLYFIN_LOG_DIR=/var/log/jellyfin
    volumes:
      - /etc/jellyfin:/etc/jellyfin
      - /var/cache/jellyfin:/var/cache/jellyfin
      - /var/lib/jellyfin:/var/lib/jellyfin
      - /var/log/jellyfin:/var/log/jellyfin
      - <path-to-media>:<path-to-media>
```

Migrating From Emby 3.5.2 to Jellyfin

> **IMPORTANT**
>
> Direct database migration from Emby (of any version) to Jellyfin is NOT SUPPORTED. We have found many subtle bugs due to the inconsistent database schemas that result from trying to do this, and strongly recommend that all Jellyfin users migrating from Emby start with a fresh database and library scan.

The original procedure is provided below for reference however we cannot support it nor guarantee that a system upgraded in this way will work properly, if at all. If anyone is interested in writing a database migration script which will correct the deficiencies in the existing database and properly import them into Jellyfin, we would welcome it however!

> **WARNING**
>
> While it is technically possible to migrate existing configuration of Emby version 3.5.2 or earlier, due to subtle and weird bugs reported after such attempts we do not recommend this migration. Emby versions 3.5.3 or 3.6+ cannot be migrated. Thus we recommend creating a new Jellyfin configuration and rebuilding your library instead.

Windows users may take advantage of the `install-jellyfin.ps1` script in the Jellyfin repository which includes an automatic upgrade option.

This procedure is written for Debian-based Linux distributions, but can be translated to other platforms by following the same general principles.

1. Upgrade to Emby version 3.5.2, so that the database schema is fully up-to-date and consistent. While this is not required, it can help reduce the possibility of obscure bugs in the database.
2. Stop the `emby-server` daemon:

   ```
   sudo service emby-server stop
   ```

3. Move your existing Emby data directory out of the way:

   ```
   sudo mv /var/lib/emby /var/lib/emby.backup
   ```

4. Remove or purge the `emby-server` package:

```
sudo apt purge emby-server
```

5. Install the `jellyfin` package using the installation instructions.
6. Stop the `jellyfin` daemon:

```
sudo service jellyfin stop
```

7. Copy over all the data files from the Emby backup data directory:

```
sudo cp -a /var/lib/emby.backup/* /var/lib/jellyfin/
```

8. Correct ownership on the new data directory:

```
sudo chown -R jellyfin:jellyfin /var/lib/jellyfin
```

9. Mark Startup Wizard as completed - if not marked as completed then it can be a security risk especially if remote access is enabled:

```
sudo sed -i '/IsStartupWizardCompleted/s/false/true/' /etc/jellyfin/system.xml
```

10. Start the `jellyfin` daemon:

```
sudo service jellyfin start
```

Hardware Acceleration

Jellyfin supports hardware acceleration (HWA) of video encoding/decoding using FFMpeg. FFMpeg and Jellyfin can support multiple hardware acceleration implementations such as Intel Quicksync (QSV), AMD AMF and NVIDIA NVENC/NVDEC through Video Acceleration APIs.

- VA-API is a Video Acceleration API that uses libva to interface with local drivers to provide HWA.
- QSV uses a modified (forked) version of VA-API and interfaces it with libmfx and their proprietary drivers (list of supported processors for QSV).

OS	Recommended HW Acceleration
Linux	QSV, NVENC, AMF, VA-API
Windows	QSV, NVENC, AMF
MacOS	VideoToolbox
RPi	V4L2

Graphics Cards comparison using HWA

Based on hardware vendor:

Vendor	Supported HW Acceleration
NVIDIA	NVENC
AMD	AMF, VA-API
Intel	QSV, VA-API
Apple	VideoToolbox
RPi	V4L2

Enabling Hardware Acceleration

Hardware acceleration options can be found in the Admin Dashboard under the **Transcoding** section of the **Playback** tab. Select a valid hardware acceleration option from the drop-down menu, indicate a device if applicable, and check `Enable hardware encoding` to enable encoding as well as decoding, if your hardware supports this.

The hardware acceleration is available immediately for media playback. No server restart is required.

On Linux you can check available GPU using:

```
lspci -nn | egrep -i "3d|display|vga"
```

or using `lshw`:

```
lshw -C display
```

H.264 / AVC 10-bit videos

The hardware decoding of H.264 10-bit aka High10 profile video is not supported by any Intel, AMD or NVIDIA GPU.

Please consider upgrading these videos to HEVC 10-bit aka Main10 profile if you want to offload your CPU usage during transcoding.

Intel Gen9 and Gen11+ iGPUs

> **NOTE**
>
> The Intel Guc/Huc firmware must be enabled for optional Low-Power encoding (pre-Gen11 only supports Low-Power H.264).

Instructions:

- ArchLinux: Arch Wiki
- Debian/Ubuntu: Brainiarc7's gist

> **WARNING**
>
> For **Jasper Lake** and **Elkhart Lake** chips (such as `N5095`, `N6005` and `J6412`), Low-Power encoding **must** be enabled. There's a known kernel issue on these chips in **linux 5.15** that comes with Ubuntu 22.04 LTS preventing you from using Low-Power. You may need to upgrade kernel for this. The linux-firmware support is **not included** in Ubuntu 20.04.3 LTS. Any Ubuntu from 21.10 **does include** the required drivers.

Supported Acceleration Methods

IMPORTANT

In Jellyfin 10.8 full hardware-accelerated filtering (scaling, deinterlacing, tone-mapping and subtitle burn-in) on Intel, AMD and NVIDIA hardware are available.

jellyfin-ffmpeg version 4.4.1-2 or higher is required, using an older or original version of FFmpeg may disable some hardware filtering improvements.

VA-API

NOTE

Intel iGPU and AMD GPU only.

A List of supported codecs for VA-API can be found on the Archlinux wiki.

WARNING

As of **Jellyfin 10.8** the official Docker image uses Debian 11 which has a compatible version of Mesa for **AMD GPU HEVC** decoding.

Earlier images do not provide a compatible version of Mesa.

Hardware acceleration on Raspberry Pi 3 and 4

WARNING

As of **Jellyfin 10.8** hardware acceleration on Raspberry Pi via `OpenMAX OMX` was dropped and is no longer available.

This decision was made because Raspberry Pi is currently migrating to a `V4L2` based hardware acceleration, which is already available in Jellyfin but does not support all features other hardware acceleration methods provide due to lacking support in FFmpeg. Jellyfin will fallback to software de- and encoding for those usecases.

The current state of hardware acceleration support in FFmpeg can be checked on the rpi-ffmpeg repository.

NVIDIA NVENC

> **NOTE**
>
> **Minimum required driver version since Jellyfin 10.8:**
>
> - **Linux:** 470.57.02
> - **Windows:** 471.41

Not every card has been tested.

If you want more than three parallel transcoding streams on a consumer (non-Quadro) NVIDIA card, you can use this patch to remove the limit. The patch is recommended for Linux and Windows but may break in the future, so check the compatible driver versions before applying it.

On Linux use `nvidia-smi` to check driver and GPU card version.

Useful links:

- Official list of supported codecs for recent NVIDIA Graphics Cards.
- Official NVIDIA ffmpeg development docs.

AMD AMF

> **NOTE**
>
> AMF is available on Windows and Linux.

> **WARNING**
>
> As of **Jellyfin 10.8** full OpenCL based hardware filtering in AMF is supported on Windows 10 and newer.
>
> AMD has not implemented the Vulkan based HW decoder and scaler in ffmpeg, the decoding speed may not be as expected on Linux.
>
> The closed source driver `amdgpu-pro` is required when using AMF on Linux.

> **TIP**
>
> Most Zen CPUs **do not** come with integrated graphics. You will need a **dedicated GPU** (dGPU) or a Zen CPU with integrated graphics for hardware acceleration. If your Zen CPU is suffixed with a *G* or *GE* in model name, you have integrated graphics.

Intel QuickSync

> **NOTE**
>
> Intel QuickSync (QSV) is derived from VA-API on Linux and D3D11VA on Windows, which can utilize Intel's fixed function hardware and EU(execution units) to do video encoding, decoding and processing.

> **IMPORTANT**
>
> To use QSV on Linux with recent Intel iGPUs the **nonfree** Intel media driver is required for full hardware acceleration. If you are using `jellyfin-ffmpeg` version 4.4.1-2 or higher it is included and you do not need to install it seperatly. Broadwell or newer generation is required for QSV on Linux, otherwise you have to use VA-API.

Useful links:

- Official list of supported codecs for recent Intel Graphics Cards.
- Intel QSV Benchmarks on Linux
- Collection of useful links and information

> **TIP**
>
> If your Jellyfin server does not support hardware acceleration, but you have another machine that does, you can leverage rffmpeg to delegate the transcoding to another machine. Currently Linux-only and requires SSH between the machines, as well as shared storage both for media and for the Jellyfin data directory.

Common setups

Each hardware acceleration type, as well as each Jellyfin installation type, has different prerequisites for enabling hardware acceleration. It is always best to consult the FFMpeg documentation on the acceleration type you choose for the latest information.

Hardware acceleration on Docker (Linux)

> **NOTE**
> This are general instructions, for more specific instructions pleas check the next sections!

In order to use hardware acceleration in Docker, the devices must be passed to the container. To see what video devices are available, you can run `sudo lshw -c video` or `vainfo` on your machine. VA-API may require the `render` group added to the docker permissions. The `render` group id can be discovered in `/etc/group` such as `render:x:122:`.

You can use `docker run` to start the server with the required permissions and devices. An example command is shown below.

```
docker run -d \
 --volume /path/to/config:/config \
 --volume /path/to/cache:/cache \
 --volume /path/to/media:/media \
 --user 1000:1000 \
 --group-add=122 \ # Change this to match your system
 --net=host \
 --restart=unless-stopped \
 --device /dev/dri/renderD128:/dev/dri/renderD128 \
 --device /dev/dri/card0:/dev/dri/card0 \
 jellyfin/jellyfin
```

Alternatively, you can use docker-compose with a configuration file so you don't need to run a long command every time you restart your server.

```
version: "3"
services:
  jellyfin:
    image: jellyfin/jellyfin
    user: 1000:1000
    group_add:
      - 122
    network_mode: "host"
    volumes:
```

```
      - /path/to/config:/config
      - /path/to/cache:/cache
      - /path/to/media:/media
    devices:
      # VAAPI Devices (examples)
      - /dev/dri/renderD128:/dev/dri/renderD128
      - /dev/dri/card0:/dev/dri/card0
```

NVIDIA hardware acceleration on Docker (Linux)

In order to achieve hardware acceleration using Docker, several steps are required.

Prerequisites:

- GNU/Linux x86_64 with kernel version > 3.10
- Docker >= 19.03
- NVIDIA GPU with Architecture > Fermi (2.1)
- NVIDIA drivers >= 361.93
- NVIDIA Container Toolkit needs to be installed

Follow the instructions in the link above to install the NVIDIA Container Toolkit for your Linux distribution.

Start your container by adding this parameter:

```
--gpus all \
```

A complete run command would look like this:

```
docker run -d \
 --name=jellyfin \
 --gpus all \
 -p 8096:8096 \
 -p 8920:8920 \
 -v /config:/config \
 -v /media:/media \
 -v /cache:/cache \
 --restart unless-stopped \
 jellyfin/jellyfin
```

Or with docker-compose >1.28, add the `deploy` section to your Jellyfin service:

```
services:
  jellyfin:
    image: jellyfin/jellyfin
    # ... your Jellyfin config
    deploy:
      resources:
        reservations:
          devices:
            - capabilities: [gpu]
```

There are some special steps when running with the following option:

```
--user 1000:1000
```

You may need to add this user to the video group on your host machine:

```
usermod -aG video <user>
```

Once the container is started you can again validate access to the host resources:

```
docker exec -it jellyfin nvidia-smi
```

If you get driver information, everything is fine but if you get an error like `couldn't find libnvidia-ml.so library in your system` you need to run the following command:

```
docker exec -it jellyfin ldconfig
```

After that, you should ensure the NVIDIA driver loads correctly.

> **NOTE**
> The official Jellyfin Docker image already sets the required environment variables to allow access to the GPUs via the NVIDIA container runtime. If you are building your own image don't forget to include `NVIDIA_DRIVER_CAPABILITIES=all` and `NVIDIA_VISIBLE_DEVICES=all` into your container's environment.

VA-API hardware acceleration on Debian/Ubuntu

Configuring VA-API on Debian/Ubuntu requires some additional configuration to ensure permissions are correct.

1. Configure VA-API for your system by following the documentation of your OS and/or vendor. Verify that a `render` device is now present in `/dev/dri`, and note the permissions and group available to write to it, in this case `render`:

   ```
   $ ls -l /dev/dri
   total 0
   drwxr-xr-x 2 root root        100 Apr 13 16:37 by-path
   crw-rw---- 1 root video  226,   0 Apr 13 16:37 card0
   crw-rw---- 1 root video  226,   1 Apr 13 16:37 card1
   crw-rw---- 1 root render 226, 128 Apr 13 16:37 renderD128
   ```

   > **NOTE**
   > On some releases, the group may be `video` or `input` instead of `render`.

2. Make sure that `jellyfin-ffmpeg` version 4.4.1-2 or higher is installed.

3. Check the output of `/usr/lib/jellyfin-ffmpeg/vainfo`.

4. Add the Jellyfin service user to the above group to allow Jellyfin's FFMpeg process access to the device, and restart Jellyfin.

   ```
   sudo usermod -aG render jellyfin
   sudo systemctl restart jellyfin
   ```

5. Configure VA-API acceleration in the `Transcoding` page of the Admin Dashboard. Enter the `/dev/dri/renderD128` device above as the `VA API Device` value.

6. Watch a movie, and verify that transcoding is occurring by watching the `ffmpeg-transcode-*.txt` logs under `/var/log/jellyfin` and using `radeontop` (AMD only) or similar tools.

Intel QuickSync (QSV) hardware acceleration on Debian/Ubuntu

1. QSV is based on VA-API device on Linux, so please confirm whether you have completed the VA-API configuration first.

2. Make sure that `jellyfin-ffmpeg` version 4.4.1-2 or higher is installed (it ships the current version of `intel-media-driver (iHD)` which is required for QSV).

3. Verify that the iHD driver is properly loaded and recognizes your iGPU.

```
sudo /usr/lib/jellyfin-ffmpeg/vainfo | grep iHD
```

4. Configure QSV acceleration in the `Transcoding` page of the Admin Dashboard.

5. Watch a movie, and verify that transcoding is occurring by watching the `ffmpeg-transcode-*.txt` logs under `/var/log/jellyfin` and using `intel_gpu_top` (can be installed with the `intel-gpu-tools` package).

VA-API and QSV hardware acceleration on LXC or LXD container

> **WARNING**
> This has been tested with LXC 3.0 and may or may not work with older versions.

Follow the steps above to add the jellyfin user to the `video` or `render` group, depending on your circumstances.

1. Install the required drivers on the host OS

2. Add your GPU to the container.

```
lxc config device add <container name> gpu gpu gid=<gid of your video or render group>
```

3. Make sure you have the required devices within the container:

```
$ lxc exec jellyfin -- ls -l /dev/dri
total 0
crw-rw---- 1 root video 226,   0 Jun  4 02:13 card0
crw-rw---- 1 root video 226,   0 Jun  4 02:13 controlD64
crw-rw---- 1 root video 226, 128 Jun  4 02:13 renderD128
```

4. Configure Jellyfin to use video acceleration and point it at the right device if the default option is wrong.

5. Try and play a video that requires transcoding and run the following, you should get a hit.

```
ps aux | grep ffmpeg | grep accel
```

6. You can also try playing a video that requires transcoding, and if it plays you're good.

Useful resources:

- LXD Documentation - GPU instance configuration
- NVIDIA CUDA inside a LXD container

VA-API and QSV hardware acceleration on LXC on Proxmox

> **IMPORTANT**
>
> Jellyfin needs to run in a **privileged** LXC container. You can convert an existing unprivileged container to a privileged container by taking a backup and restoring it as priviledged.

1. Install the required drivers on the Proxmox host
2. Add your GPU to the container by editing `/etc/pve/lxc/<container-id>.conf` (you may need to change the GIDs in the examples below to match those used on you host).

   > **WARNING**
   >
   > This has been tested on `Proxmox VE 7.1` - on previous versions you may need to change `cgroup2` to `cgroup`.

   Intel iGPU:

   ```
   lxc.cgroup2.devices.allow: c 226:0 rwm
   lxc.cgroup2.devices.allow: c 226:128 rwm
   lxc.mount.entry: /dev/dri/card0 dev/dri/card0 none bind,optional,create=file
   lxc.mount.entry: /dev/dri/renderD128 dev/dri/renderD128 none bind,optional,create=file
   ```

   NVidia GPU:

   ```
   lxc.cgroup2.devices.allow: c 195:* rwm
   lxc.cgroup2.devices.allow: c 243:* rwm
   lxc.mount.entry: /dev/nvidia0 dev/nvidia0 none bind,optional,create=file
   lxc.mount.entry: /dev/nvidiactl dev/nvidiactl none bind,optional,create=file
   lxc.mount.entry: /dev/nvidia-uvm dev/nvidia-uvm none bind,optional,create=file
   lxc.mount.entry: /dev/nvidia-modeset dev/nvidia-modeset none bind,optional,create=file
   lxc.mount.entry: /dev/nvidia-uvm-tools dev/nvidia-uvm-tools none bind,optional,create=file
   ```

3. Shutdown and start your container.

4. Install the required drivers in your container.
5. Add the jellyfin user to the `video`, `render` and/or `input` groups depending on who owns the device inside the container.
6. Configure Jellyfin to use hardware acceleration and point it at the right device if the default option is wrong.
7. Try and play a video that requires transcoding and run the following, you should get a hit.

```
ps aux | grep ffmpeg | grep accel
```

8. You can also try playing a video that requires transcoding, and if it plays you're good.

AMD AMF encoding on Ubuntu 18.04 or 20.04 LTS

1. Install the `amdgpu-pro` closed source graphics driver by following the installation instructions.
2. Then install `amf-amdgpu-pro`.

```
sudo apt install amf-amdgpu-pro
```

3. Check if `jellyfin-ffmpeg` contains `h264_amf` encoder:

```
$ cd /usr/lib/jellyfin-ffmpeg/
$ ./ffmpeg -encoders | grep h264_amf
V..... h264_amf             AMD AMF H.264 Encoder (codec h264)
```

> **NOTE**
> If not available, update your `jellyfin-ffmpeg` to the latest version and try again.

4. Choose AMD AMF video acceleration in Jellyfin and check the `Enable hardware encoding` option.
5. Watch a movie, then verify that `h264_amf` encoder is working by watching the `ffmpeg-transcode-*.txt` transcoding logs under `/var/log/jellyfin` and using `radeontop` or similar tools.

AMD AMF encoding on Arch Linux

AMD does not provide official `amdgpu-pro` driver support for Arch Linux, but fortunately, a third-party packaged `amdgpu-pro-installer` is provided in the archlinux user repository.

1. Clone this repository using `git`.

```
git clone https://aur.archlinux.org/amdgpu-pro-installer.git
```

2. Enter that folder and make the installation package and install it.

```
cd amdgpu-pro-installer
makepkg -si
```

3. Go to step 3 of Configuring AMD AMF encoding on Ubuntu 18.04 or 20.04 LTS above.

OpenCL / CUDA / Intel VPP Tone-Mapping

Hardware based HDR10/HLG/DoVi tone-mapping with NVIDIA NVENC, AMD AMF, Intel QSV and VA-API is done through OpenCL or CUDA. DoVi Profile 5 and 8 tone-mapping requires `jellyfin-ffmpeg` version 5.0.1-5 or higher.

Intel hardware based VPP HDR10 tone-mapping is supported on Intel QSV and VA-API on Linux. VPP is prefered when both two tone-mapping options are checked on Intel.

OS/Platform	NVIDIA NVENC	AMD AMF	Intel QSV	Intel VA-API	AMD VA-API	Software
Linux	✓	✓	✓	✓	✓	WIP
Windows	✓	✓	✓	N/A	N/A	WIP
Docker	✓	✓	✓	✓	✓	WIP

> **NOTE**
> Tone-mapping on Windows with Intel QSV and AMD AMF requires Windows 10 or newer.

> **IMPORTANT**
> Make sure the hardware acceleration is well configured before configuring tone-mapping with this instructions.

1. **On Windows:** Install the latest NVIDIA, AMD or Intel drivers.

2. **On Linux or Docker:**

- For **NVIDIA cards** no further configuration is necessary.
- For **AMD cards**, install `amdgpu-pro` with opencl arguments (see Configuring AMD AMF encoding on Ubuntu 18.04 or 20.04 LTS for more details):

```
sudo ./amdgpu-pro-install -y --opencl=pal,legacy
sudo usermod -aG video $LOGNAME
sudo usermod -aG render $LOGNAME
```

- For **Intel iGPUs**, you have two types of tone-mapping methods: OpenCL and VPP. The latter one does not support fine tuning options.

OpenCL: Follow the instructions from intel-compute-runtime. If you are using the official Docker image or the one from linuxserver this step can be skipped.

VPP: Make sure `jellyfin-ffmpeg` 4.4.1-2 or higher is installed. Previous versions did not ship `intel-media-driver` thus it was required to be installed manually.

- When running on docker, the **privileged** flag is required for the OpenCL device to be recognized. You can do this by adding `--privileged` to your docker command or `privileged: true` to your docker compose file.

> **WARNING**
>
> Tone-mapping on Intel VA-API and QSV **requires an iGPU that supports 10-bit decoding**, such as i3-7100 or J4105.

> **IMPORTANT**
>
> Do **not use** the `intel-opencl-icd` package from your distro's repository since they were not built with `RELEASE_WITH_REGKEYS` enabled, which is required for P010 pixel interop flags.

3. **Debugging:** Check the OpenCL device status. You will see corresponding vendor name if it goes well.

   - Use `clinfo`: Install `clinfo` before using it. `sudo apt install -y clinfo` on Debian/Ubuntu or `sudo pacman -Sy clinfo` on Arch. Then `sudo clinfo`.

   - Use `jellyfin-ffmpeg`: `/usr/lib/jellyfin-ffmpeg/ffmpeg -v debug -init_hw_device opencl`

Verifying Transcodes

To verify that you are using the proper libraries, run this command against your transcoding log. This can be found at `Admin Dashboard > Logs`, and `/var/log/jellyfin` if installed via the apt repository.

```
grep -A2 'Stream mapping:' /var/log/jellyfin/ffmpeg-transcode-<random-id>>.log
```

This returned the following results.

```
...
Stream mapping:
Stream #0:0 -> #0:0 (hevc (native) -> h264 (h264_qsv))
Stream #0:1 -> #0:1 (aac (native) -> mp3 (libmp3lame))
...
```

`Stream #0:0` used software (VAAPI Decode can also say native) to decode HEVC and used HWA to encode.

```
...
Stream mapping:
Stream #0:0 -> #0:0 (h264 (hevc_qsv) -> h264 (h264_qsv))
Stream #0:1 -> #0:1 (flac (native) -> mp3 (libmp3lame))
...
```

`Stream #0:0` used HWA for both. `hevc_qsv` to decode and `h264_qsv` to encode.

Troubleshooting

This page outlines some solutions to common issues beginners may encounter when running Hello World application.

Issues

The easiest way to check for issues is by checking the logs, which can be accessed through the console for the web client or in the log directory on your server.

You can check the video below to fix the issue

Networking Issues

If you can access the web interface over HTTP but not HTTPS, then you likely have an error with the certificate. HelloWorld uses a PFX file to handle HTTPS traffic. If you created the file with a password, then you will have to enter that value on the **Networking** page in the settings.

If you can access the server locally but not outside of your LAN, then you likely have an issue with the router configuration. Check the port forwarding settings on your router to ensure the server is visible from outside your local network. You can also enable the "Enable automatic port mapping" option on the **Networking** page of the server settings to have the server attempt to configure port forwarding on the router automatically if your router supports it.

If there are no logs at all relating to web traffic, even over a LAN connection, then the server hasn't been reached at all yet. This would indicate either an incorrect address or an issue somewhere else on the network.

Debug Logging

To enable debug (much more verbose) logging, it is currently required to manually edit config files since no options exist yet on the frontend. Go to the HelloWorld configuration directory, find the `logging.default.json` file, and change the minimum level to debug as seen below.

```
{
    "Serilog": {
        "MinimumLevel": {
            "Default": "Debug"
        }
    }
}
```

HelloWorld will automatically reload the new configuration without needing to restart. The debug messages show up in the log with the `DBG` tag.

Real Time Monitoring

This will let HelloWorld automatically update libraries when files are added or modified. Unfortunately this feature is only supported on certain filesystems.

For Linux systems, this is performed by inotify. NFS and rclone do not support inotify, but support can be provided by using a union file system such as mergerfs with your networked file systems.

Due to the library size, you can receive an error such as this:

```
[2019-12-31 09:11:36.652 -05:00]  [ERR] Error in Directory watcher for: "/media/movies"  System.
```

If you are running Debian, RedHat, or another similar Linux distribution, run the following in a terminal:

```
echo fs.inotify.max_user_watches=524288 | sudo tee -a /etc/sysctl.d/40-max-user-watches.conf
&& sudo sysctl -p
```

If you are running ArchLinux, run the following command instead:

```
echo fs.inotify.max_user_watches=524288 | sudo tee /etc/sysctl.d/40-max-user-watches.conf &&
sudo sysctl --system
```

Then paste it in your terminal and press on enter to run it. For Docker, this needs to be done on the host, not the container. See here for more information.

Uninstall MacOS

To fully remove all data of HelloWorld from MacOS, run these commands:

```
rm -Rfv ~/.config/HelloWorld
rm -Rfv ~/.cache/HelloWorld
rm -Rfv ~/.local/share/HelloWorld
```

Unlock locked user account

When the admin account is locked out and the Forgot Password feature is not working, you have to unlock the user manually. To do that, you need to find the `HelloWorld.db` file on your system. The default location on Linux is: `/var/lib/HelloWorld/data/`. For paths in other environments, see server paths.

Linux CLI

Before continuing, make sure that you have sqlite3 installed. When sqlite3 is not installed, you can install it under Debian based systems with `apt install sqlite3`. After that do the following commands/SQL query:

```
sqlite3 /PATH/TO/HelloWorld/DB/HelloWorld.db
```

```
UPDATE Users SET InvalidLoginAttemptCount = 0 WHERE Username = 'LockedUserName';
UPDATE Permissions SET Value = 0 WHERE Kind = 2 AND UserId IN (SELECT Id FROM Users WHERE
Username = 'LockedUserName');
.exit
```

SQLiteBrowser

It is also possible to use SQLiteBrowser on systems with a desktop environment. Start by opening the database inside the SQLite Browser. After opening the database, navigate to the Execute SQL Tab and execute the following query:

```
UPDATE Users SET InvalidLoginAttemptCount = 0 WHERE Username = 'LockedUserName';
UPDATE Permissions SET Value = 0 WHERE Kind = 2 AND UserId IN (SELECT Id FROM Users WHERE
Username = 'LockedUserName');
```